



VNIVERSITAT
DE VALÈNCIA

PERFORMANCE OPTIMISATION OF BIOLOGICAL PATHWAY DATA STORAGE, RETRIEVAL, ANALYSIS AND ITS INTERACTIVE VISUALISATION

Ph.D. Thesis by:
Antonio Fabregat Mundo

Supervisors:
Dr. Pablo Marín García
Dr. Vicente Arnau Llombart

Programa de Doctorat en Tecnologies de la
Informació, Comunicacions i Computació

Departament d'Informàtica

Escola Tècnica Superior d'Enginyeria

Universitat de València

April 2018

*To my beloved wife and daughter;
I really do hope weekends will be ours from now on!*

ACKNOWLEDGMENTS

I am very grateful to both of my supervisors, Pablo Marín and Vicente Arnau for their valuable input and guidance. Foremost, I would like to thank Pablo for convincing me, in the first instance, to pursue a Ph.D. and for being supportive ever since the beginning of this process.

I am also grateful to many more people; to Henning Hermjakob who has provided with the adequate environment that allowed me to develop this work in the Reactome team at the European Bioinformatics Institute (EMBL-EBI). To my work colleagues; Konstantinos Sidiropoulos, Guilherme Viteri and Florian Korninger with whom I had the pleasure of sharing part of some projects; and to Steve Jupe and Bijay Jassal for their constructive comments and for the help in correcting this manuscript.

I would also like to express my gratitude to Professor Manuel Forner, who has always been of great support and a source of inspiration. Even though he has not had a direct role in this thesis, I could have never reached the point of starting it without his help in the past.

Last but not least, my special thanks go to my wife and daughter for being so patient with me and to everyone else in my family: my parents, sister, brother in law and niece for their trust and understanding during all these years.

ABSTRACT

The aim of this research was to optimise the performance of the storage, retrieval, analysis and interactive visualisation of biomolecular pathways data. This was achieved by the adoption of new technologies and a variety of highly optimised data structures, algorithms and strategies across the different layers of the software.

The first challenge to overcome was the creation of a long-lasting, large-scale web application to enable pathways navigation; the Pathway Browser. This tool had to aggregate different modules to allow users to browse pathway content and use their own data to perform pathway analysis.

Another challenge was the development of a high-performance pathway analysis tool to enable the analysis of genome-wide datasets within seconds. Once developed, it was also integrated into the Pathway Browser allowing interactive exploration and analysis of high throughput data.

The Pathways Overview layout and widget were created to enable the representation of the complex parent-child relationships present in the pathways hierarchical organisation. This module provides a means to overlay analysis results in such a way that the user can easily distinguish the most significant areas of biology represented in their data. Although an existing force-directed layout algorithm was initially utilised for the graphical representation, it did not achieve the expected results and a custom radial layout algorithm was developed instead.

A new version of the pathway Diagram Viewer was engineered to achieve loading and rendering of 97% of the target diagrams in less than 1 second. Combining the multi-layer HTML5 Canvas strategy with a space partitioning

data structure minimised CPU workload, enabling the introduction of new features that further enhance user experience.

On the server side, the work focused on the adoption of a graph database (Neo4j) and the creation of the new Content Service (REST API) that provides access to these data. The Neo4j graph database and its query language, Cypher, enabled efficient access to the complex pathway data model, facilitating easy traversal and knowledge discovery. The adoption of this technology greatly improved query efficiency, reducing the average query time by 93%.

ABBREVIATIONS

Abbreviation	Description
ACID	Atomicity, Consistency, Isolation, and Durability
ADP	Adenosine Diphosphate
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATP	Adenosine Triphosphate
CC-BY	Creative Commons Attribution license
CGI	Common Gateway Interface
CORS	Cross-Origin Resource Sharing
CPU	Central Processing Unit
DDR3	Double Data Rate type three SDRAM (DDR3 SDRAM)
DIV	Division or Section (HTML tag)
DNA	Deoxyribonucleic Acid
DOM	Document Object Model
EHL	Enhanced High Level Diagram
EMF	Enhanced MetaFile
FDR	False Discovery Rate
GB	Gigabyte
GHz	Gigahertz
GO	Gene Ontology
GWT	Google Web Toolkit

HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LRU	Least Recently Used
MHz	Megahertz
MVC	Model View Controller
MVP	Model View Presenter
NP	Nondeterministic Polynomial
ORA	Over-representation Analysis
ORM	Object-Relational Mapping
PAC	Presenter Abstraction Control
PDB	Pathway Database
PE	Physical Entity
PNG	Portable Network Graphics
PPTX	PowerPoint XML format
PSI	Proteomics Standards Initiative
PSICQUIC	Proteomics Standards Initiative Common Query Interface
RAP	Release Automation Pipeline
REST	Representational State Transfer
RLE	ReactionLikeEvent
RNA	Ribonucleic Acid
SBGN	Systems Biology Graphical Notation
SDN	Spring Data Neo4j
SoC	Separation of Concerns

SOP	Same-Origin Policy
SPA	Single Page Application
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TPS	Transactions Per Second
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
W3C	World Wide Web Consortium
XML	Extensible Markup Language
YUI	Yahoo User Interface
ZUI	Zoomable User Interface

Table of contents

Acknowledgments	i
Abstract	iii
Abbreviations	v
1. Introduction	1
1.1. Background.....	1
1.2. About Reactome.....	5
1.2.1. <i>Curation</i>	8
1.2.2. <i>Software</i>	9
1.3. Aim and objectives.....	11
1.4. References.....	12
2. Pathway Browser	17
2.1. Reactome Pathway Browser state of the art.....	18
2.2. Re-engineering the Pathway Browser.....	20
2.2.1. <i>Software Layout Pattern. Model View Presenter (MVP)</i>	21
2.2.2. <i>Event Bus</i>	24
2.2.3. <i>State Manager</i>	25
2.2.4. <i>Memory Consumption</i>	27
2.2.5. <i>Widgets for third party data integration</i>	30
2.2.6. <i>Resulting web application</i>	33
2.3. Summary.....	35
2.4. References.....	37
3. Pathway Analysis Tools	39
3.1. The relational databases approach.....	40
3.2. A high-performance in-memory approach.....	41

3.2.1. <i>New software ecosystem</i>	45
3.2.2. <i>Lightweight client integrated into the Pathway Browser</i>	48
3.2.3. <i>Overlaying analysis results in the Pathway Browser</i>	49
3.3. Summary.....	51
3.4. References	52
4. Pathways Overview	55
4.1. Finding an appropriate layout.....	56
4.1.1. <i>Force-directed layout approach</i>	57
4.1.2. <i>Layout requirement</i>	60
4.1.3. <i>Custom radial layout definition</i>	61
4.1.4. <i>Storing the result of the custom radial layout</i>	75
4.2. Lightweight web browser widget.....	77
4.2.1. <i>Widget details</i>	77
4.2.2. <i>Overlaying analysis results</i>	81
4.3. Summary.....	84
4.4. References	85
5. Diagram Viewer	89
5.1. Evolution of the previous versions of the Diagram Viewer	90
5.1.1. <i>Diagram Viewer version 1</i>	92
5.1.2. <i>Diagram Viewer version 2</i>	97
5.2. Diagram Viewer version 3.....	100
5.2.1. <i>Data structures and strategies to boost performance</i>	101
5.2.2. <i>Text book style illustrations: Enhanced High Level Diagrams</i> ..	111
5.2.3. <i>Overlaying analysis results</i>	116
5.2.4. <i>Export options</i>	119
5.2.5. <i>Results and discussion</i>	121
5.3. Summary.....	124

5.4. References	126
6. The graph database	129
6.1. Reactome data model	130
6.2. Moving from a relational to a graph database	131
6.2.1. <i>The graph importer</i>	132
6.2.2. <i>Data integrity, consistency and quality assessment</i>	137
6.2.3. <i>New software ecosystem based on the graph database</i>	141
6.2.4. <i>Results and discussion</i>	145
6.3. Summary	151
6.4. References	152
7. Conclusions	157
A. Queries for data quality assessment	161
P. Publications directly related to the presented work	165
P.1. Reactome pathway analysis: a high-performance in-memory approach.....	165
P.2. Reactome diagram viewer: Data structures and strategies to boost performance	175
P.3. Reactome enhanced pathway visualisation.....	183
P.4. Reactome Graph Database: Efficient access to complex pathway data.....	191
P.5. The Reactome pathway knowledgebase (NAR 2017)	205
P.6. The Reactome pathway knowledgebase (NAR 2015)	213
P.7. The Reactome pathway knowledgebase (NAR 2013)	221
O. Publications indirectly related to the presented work	227
O.1. Gramene 2018: unifying comparative genomics and pathway resources for plant research	227

0.2. Open Targets: a platform for therapeutic target identification and validation	228
0.3. Plant Reactome: a resource for plant pathways and comparative analysis.....	229
0.4. Gramene 2016: comparative plant genomics and pathway resources	230

Figures index

Figure 1.1. Reactions and pathways	2
Figure 1.2. Schematic view of the curation process.....	9
Figure 2.1. Event Browser. Reactome’s early tool for accessing its content	19
Figure 2.2. The first version of the Reactome Pathway Browser.....	19
Figure 2.3. The position and role of each of the three main actors in the MVP pattern.	23
Figure 2.4. Event bus with the different modules and managers.	24
Figure 2.5. The traditional page life cycle versus the SPA life cycle.....	25
Figure 2.6. Pathway Browser main viewport and “Structures” tab.....	30
Figure 2.7. Pathway Browser main viewport and “Expression” tab	32
Figure 2.8. Pathway Browser illustration highlighting the different panels	34
Figure 3.1. Representation of two analysis use cases joining the different data structures.....	44
Figure 3.2. Analysis Service ecosystem.....	46
Figure 3.3. Reactome Analysis Service RESTful API documentation page	47
Figure 3.4. The Analysis tools data submission interface	49
Figure 3.5. Analysis results for a PRIDE dataset.....	50
Figure 4.1. A reaction map view called “Starry Sky”	55
Figure 4.2. Force Atlas layout algorithm applied to the human pathways in Reactome	58

Figure 4.3. Force Atlas layout for a duration of 10 minutes	59
Figure 4.4. Hypothetical example of pathways and their relationships for scenario 1	63
Figure 4.5. Schematic representation of the construction of the customised radial layout for the first level of Figure 4.4 containing 5 hypothetical top-level-pathways.....	64
Figure 4.6. Schematic representation of the construction of the second ring of the customised radial layout	66
Figure 4.7. Represents the logic behind the calculation of centre of the node to be added to a given ring.....	68
Figure 4.8. Quadratic functions to render lines joining parent-child related nodes vs linear functions.	70
Figure 4.9. Result of applying the custom radial layout algorithm.....	70
Figure 4.10. Pathways Overview using the custom radial layout for scenario 1	71
Figure 4.11. First attempt using the custom layout algorithm per top-level pathway	73
Figure 4.12. Final representation of the Pathways Overview using the custom radial layout.....	74
Figure 4.13. UML class diagram for the Pathways Overview layout data container	75
Figure 4.14. JSON example for release 63.....	76
Figure 4.15. Pathways Overview widget representing the pathways contained in the data release version 54	78

Figure 4.16. A zoomed-in view of the Metabolism burst showing individual subpathway groups.....	79
Figure 4.17. The RAF/MAP kinase cascade pathway is highlighted to show its involvement in multiple bursts.....	80
Figure 4.18. Different types of analysis overlay for the Pathways Overview widget	82
Figure 4.19. Pathways Overview zoomed in to “Metabolism” overlaying the results of an expression analysis	83
Figure 5.1. User interface for the first version of the Reactome Event Browser. ...	89
Figure 5.2. First version of the Diagram Viewer	93
Figure 5.3. Diagram displaying the coloured physical entities that correspond to expression values of the experimental data.....	95
Figure 5.4. Protein–protein interactions displayed in the Pathway Browser	96
Figure 5.5. Pathway analysis overlay in the Diagram Viewer version 2.....	98
Figure 5.6. Example of disease curation and visualisation in Reactome.....	100
Figure 5.7. XML vs JSON comparison charts.....	103
Figure 5.8. Schematic view of a pathway made up of two reactions	105
Figure 5.9. UML sequence diagram comparing sequential and render-first loading strategies.....	107
Figure 5.10. A simplified example of the multi-layer canvas strategy.....	108
Figure 5.11. Hypothetical diagram composed of two separate reactions	109

Figure 5.12. Diagram Viewer version 3 controls the flow of displayed information by the level of zoom.....	111
Figure 5.13. Hemostasis as previously represented with subpathway icons.	112
Figure 5.14. Hemostasis as an Illustration.	113
Figure 5.15. Hemostasis as an Enhanced High Level Diagram (with the SVG). ..	114
Figure 5.16. Overrepresentation and Expression analysis overlay	117
Figure 5.17. Correspondence between the hierarchy and EHLD	119
Figure 5.18. The pathway diagram for Striated Muscle Contraction exported to Microsoft PowerPoint	120
Figure 5.19. Comparison of perceived loading times achieved by Diagram Viewer version 2 and version 3 versus the diagram size (in number of diagram entities)	122
Figure 6.1. A simplified example where reactions only contain reactants and products represented by the class PhysicalEntity	134
Figure 6.2. Representation of the content migration to the graph database.....	136
Figure 6.3. Example of data quality test written in Cypher.....	139
Figure 6.4. A schematic diagram of the new software ecosystem.....	142
Figure 6.5. Content Service documentation page.....	144
Figure 6.6. Examples of frequent use cases that can be answered using Cypher queries	145
Figure 6.7. Comparison of the response and elapsed time for one user sequentially retrieving 5,000 reaction instances from the graph and relational databases	147

- Figure 6.8.** Response time versus an increasing set of users simultaneously performing queries for 5,000 reaction instances..... 148
- Figure 6.9.** Throughput measured in transactions per second, versus the number of users concurrently performing queries for 5,000 reaction instances..... 149
- Figure 6.10.** The Reactome graph database in numbers..... 150

1. INTRODUCTION

As a computer scientist with an interest in the field of biology, my main motivation for conducting the work in this thesis was the possibility of delving deeper into biomolecular pathways data, applying the concepts learnt during my degree in computer science engineering and master in artificial intelligence with focus on data modelling.

This thesis focuses on the storage, retrieval, analysis and visualisation of biomolecular pathways data. Therefore, this introduction starts with a summary of concepts relevant to the biology of biomolecular pathways, continues with an introduction to the Reactome pathways database and finishes by setting the aims and objectives.

1.1. BACKGROUND

The basic unit of a biomolecular pathway is a reaction, typically with substrates, enzymes, and products (Figure 1.1a). A collection of connected biomolecular reactions can be considered to constitute a biological pathway. Connections are formed when an output of one reaction becomes an input for a subsequent reaction, or when that output can act as a catalyst for another reaction (Figure 1.1b).

A Pathway Database (PDB), also referred to as a “knowledge base”, is a bioinformatics database that condenses biological knowledge of molecular interactions into pathway data collections that describe defined biochemical pathways. PDBs can usually retrieve and display data from different sources, creating useful links between different databases. Most PDBs contain computable descriptions of pathways structured by using a formal ontology,

as opposed to textual or semistructured descriptions of pathways [Karp 2001; Garcia-Campos et al. 2015].

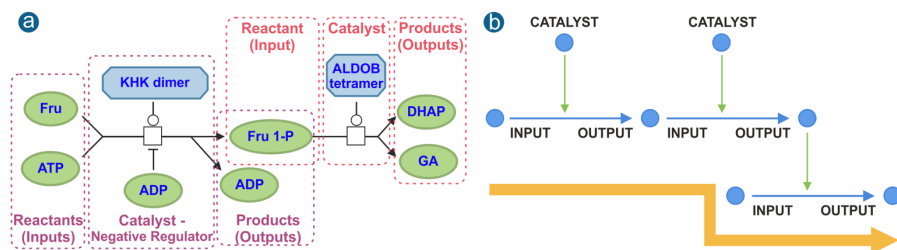


Figure 1.1. Reactions and pathways. (a) Example of a reaction containing reactants (AMP, H₂O), catalyst (NT5C1B), regulators (ADP) and products (Pi, Ade-Rib). (b) Abstract representation of a pathway (yellow arrow) as the concatenation of two or more reactions where an output of one reaction becomes an input or catalyst for a subsequent reaction.

Generally, experimental evidence from literature provides provenance for the curation of a reaction. Computational analyses can be carried out by the database itself to infer possible functions of homologous biomolecules. Additionally, cross-referencing of shared data between databases is generally performed. Currently hundreds of PDBs projects are established, each actively annotating biological knowledge in specialised contexts. The current PDB catalogue is truly abundant and diverse, differing in species focus, curation approach and the kinds of pathways and interactions they describe [Bader et al. 2006]. As the PDB catalogue has grown, it has increasingly allowed the interconnectedness between different data from different databases, improving the knowledge of biological processes for use in research and discovery [Garcia-Campos et al. 2015].

Researchers who want more information on proteins of interest can use pathway analysis tools to help them better understand or interpret ‘omics data

from PDBs. These analysis methods have applications in biomedical research where the results can help to uncover new insights into the biomolecules and processes [Garcia-Campos et al. 2015]. The data returned from pathway analysis tools should never be treated as a complete truth, it is better considered as a useful starting point that may reveal relationships within discrete biological areas.

To better understand complex biological data, enrichment techniques are key tools to reduce data complexity, improve data interpretation and help generate plausible hypotheses. Although ‘omics tools are growing and steadily improving, useful metabolomic analysis tools are scarce, most being developed from other ‘omics technologies [Marco-Ramell et al. 2018].

The most common analysis method for performing pathway analyses is functional enrichment, also called over-representation analysis (ORA). This method is a statistical test that determines whether certain molecules are over-represented (enriched) in the submitted data. Current ORA methods can provide robust, consistent and reproducible results, regardless of their analytical approach or PDB used [Marco-Ramell et al. 2018]. The overall aim of ‘omics analysis tools is to produce clear and meaningful integrated displays without showing the intrinsic complexity of the data that ultimately helps to provide biological insight [Gehlenborg et al. 2010].

The aim is to generate testable hypotheses about biological processes that govern system behaviour [Moyano et al. 2015]. Therefore, biological processes are modelled as networks of graphical maps that consist of nodes and edges, representing the individual system components and their relationships, respectively. Depending on the system components used, different biological networks can arise, such as gene regulatory networks, signalling networks, and metabolic networks.

As demonstrated by their frequent use in scientific literature and discussions, diagrams are an important communication tool for biologists. From the user's perspective, representing pathway knowledge as a diagram is a helpful, intuitive way to representing biological information and share it with others [Perini 2013]. For pathway visualisation, Le Novere et al. define three orthogonal, complementary types of diagram for the Systems Biology Graphical Notation (SBGN) that can be seen as alternative projections of complex biological information. These are the process diagram, the entity relationship diagram and the SBGN activity flow diagram. Of these types, the most appropriate method for representing Reactome pathways is the entity relationship diagram, because it emphasises the influences that entities have upon each other's transformations, rather than the transformations themselves [Le Novere et al. 2009].

The challenge of pathway data visualisation has been addressed by several resources. In cases like MINERVA [Gawron et al. 2016] and NAVICELL [Kuperstein et al. 2013] the Google map engine was adopted to visualise pathways using SBGN. WikiPathways [Kutmon et al. 2016] and KEGG [Kanehisa et al. 2014] display pathways using an in-house developed viewer. Resources such as Pathway Commons [Cerami et al. 2011] display pathways as networks of gene–gene interactions. The most common navigation features used to explore pathway diagrams are zooming, panning and selection of pathway elements to view detailed information. Some tools, such as MINERVA and WikiPathways, also allow users to map drug targets or overlay experimental data. Another popular tool for pathway analysis and visualisation is the Ingenuity Pathway Analysis tool (<https://www.qiagenbioinformatics.com>), but as a commercial tool it is inaccessible to many users.

User Experience (UX) is conceptually a dynamic, context-dependent and subjective process, which stems from the broad range of potential benefits that users may derive from a product. UX is a relatively new area that is increasingly considered to be an essential part of the human-computer interaction domain, which should now be grounded in user-centred design practices [Law et al. 2009]. Efficient performance has a positive influence on UX measurements. It has been shown that users can have immediate impressions about an application after being exposed to it for as little as 50 ms [Lindgaard et al. 2011].

1.2. ABOUT REACTOME

Reactome (<https://reactome.org>) is a free, open-source, open-data, curated and peer reviewed pathway database. Its goal is to provide intuitive bioinformatics tools for the visualisation, interpretation and analysis of pathway knowledge to support basic research, genome analysis, modelling, systems biology and education. Information in the database is authored, entered and maintained by a team of expert biologists.

Life on the cellular level is a network of molecular interactions. Molecules are synthesised and degraded, undergo a bewildering array of temporary and permanent modifications, are transported from one location to another, and form complexes with other molecules. Reactome represents all of this complexity as reactions in which input molecules are converted to output. These reactions can occur spontaneously or be facilitated by physical entities acting as catalysts, and their progress can be modulated by regulatory effects of other physical entities. Reactions are linked together by shared physical entities: a product from one reaction may be a substrate in another reaction and may catalyse yet a third. It is often convenient, if sometimes arbitrary, to group such sets of interlinked reactions into pathways.

The functions of macromolecular entities such as proteins are often determined not only by their primary sequences but by chemical modifications they have undergone. In Reactome, unmodified and modified forms of a protein are distinct physical entities and the modification process is treated as an explicit reaction. A macromolecule's function may depend on whether the molecule is free or complexed with specific other molecules. Reactome treats complexes as physical entities distinct from their components, and the multimerisation events that build up complexes are modelled explicitly as reactions.

A Physical Entity (PE) can be any individual molecule, sets of molecules or complexes grouped together on the basis of shared characteristics. In Reactome, PEs can be chemicals, proteins, DNA or RNA and sets or complexes of any of these in combination (complexes) or collection (sets). The difference between a complex and a set is that the first one represents real constructs present in biology whereas the second one is a conceptual idea to represent similar molecules that have indistinguishable roles in a reaction. A special case of set is the candidate set which describes a set of molecules that contains at least one member that might be anticipated to be functionally equivalent to the other members of the set but this has yet to be experimentally verified. Candidate sets are often used in Reactome when some members of a protein family have been demonstrated to participate in a reaction (defined as Members of the set) and other related proteins are anticipated to have similar properties but have not been proven to do so.

Many biochemical entities and processes appear redundant: there are two or more chemically distinct entities that can act more or less interchangeably. It is often useful to treat functionally equivalent protein isoforms, splice variants, and paralogues as a collection in a set, implying that any individual

entity from the given set could fulfil the same role in a given situation. The Reactome data model allows this type of generalisation, but does so explicitly in a way that allows researchers to trace specific functions back to the individual molecules covered by the generalisation.

Cellular compartments play a key role in biological processes. The segregation of molecules into different compartments often regulates the reactions in which those entities can participate, or can be responsible for driving a reaction forward. In Reactome, a molecule in one compartment is distinct from that molecule in another compartment. Thus, extracellular and cytosolic glucose are different Reactome entities and, e.g., the movement of glucose across the plasma membrane is a reaction that converts the extracellular glucose entity into the cytosolic one.

The goal of the Reactome knowledgebase is to represent human biological processes, but many of these processes have not been directly studied in humans. Rather, a human event has been inferred from experiments on material from a model organism. In such cases, the model organism reaction is annotated in Reactome, the inferred human reaction is annotated as a separate event, and the inferential link between the two reactions is explicitly noted.

Reactome utilises a frame-based knowledge representation [Vastrik et al. 2007]. The data model (<https://reactome.org/content/schema>) consists of classes (frames) that describe different concepts such as reaction or entity. Classes have attributes (slots) that hold properties of the represented class instances, e.g. names and identifiers. The value types contained in the slots can be primitive (string, numbers, or boolean) or references to other class instances. Knowledge in Reactome is captured as instances of these classes with their associated attributes.

1.2.1. CURATION

Reactome systematically links human proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for discovering unexpected functional relationships in data such as gene expression pattern surveys or somatic mutation catalogues from tumor cells. Reactome database annotations are manually curated from literature by expert biologists, and cross-referenced to many resources such as PubMed (www.ncbi.nlm.nih.gov/pubmed/), Ensembl (www.ensembl.org) [Zerbino et al. 2018], UniProt (www.uniprot.org/) [Uniprot Consortium 2017], NCBI (www.ncbi.nlm.nih.gov/) [NCBI Consortium 2018], ChEBI (www.ebi.ac.uk/chebi/) [Hastings et al. 2013], KEGG (Gene and Compound) (www.genome.jp/kegg/) [Kanehisa et al. 2016], and Gene Ontology (GO) (www.geneontology.org/) [GO Consortium 2017].

Part of the curation process is to draw the connected reactions in a pathway using the Reactome curator tool interface. Reactions can be considered as pathway ‘steps’. These diagrams, together with the curated data, is uploaded to the central database from where it will be displayed on the Reactome website.

The goal of Reactome is to describe the known biochemical details of human biological pathways. A Reactome curator's job is to work with experts in different fields of biology to identify and curate suitable human pathways by breaking them down into subpathways and reactions and describing them in a format that is compatible with the Reactome data model. There are several ways curators can start their project (Figure 1.2a). They may already have expertise in a specific biological area, recruit an external expert at conferences, jamborees and training events, recruit an expert via their help

mail they've sent to Reactome to add new material or identify an expert who wishes to collaborate with the Reactome project.

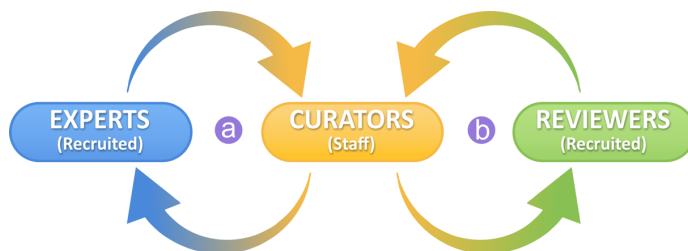


Figure 1.2. Schematic view of the curation process. (a) Reactome curators and recruited experts work together to identify, curate and layout suitable human pathways. (b) Recruited reviewers check the final result and work together with Reactome curators to ensure accuracy and correctness of the work so the curated pathway can be released.

Once recruited, an expert is able to provide the fine-grained, molecular details a Reactome curator uses to create the relevant reactions and pathways in the Reactome curator tool interface. Once a pathway has been created, it is sent to a senior curator who checks the pathway to make sure it conforms to the data-model. When that process is done, external reviewers are invited to peer-review the material to check that it conforms to a generally-accepted view of the biological area it represents. The reviewer may send feedback to the curator to revise some parts of the pathway (Figure 1.2b). When the external reviewer is completely happy with the material, the pathway is now ready for the quarterly release process for public access.

1.2.2. SOFTWARE

The Reactome software ecosystem comprises open source tools and applications that can be grouped into two main categories based on their use: (i) software tools, used internally, to support or carry out the curation and

release process and (ii) software tools and services, used by the public, to browse, visualise, search, discover, and analyse the Reactome content.

The first group includes the Curator Tool, a JAVA desktop application meant for use by curators to annotate biological pathways based on the Reactome schema, and the Release Automation Pipeline (RAP), an aggregation of Perl and Shell scripts that automate and coordinate the execution of all the necessary steps before the curated content is released to the public. It should be noted that knowledge is stored in a MySQL relational database and accessed via the GKInstance library, a custom Object-Relational Mapping (ORM) and persistence layer that handles all database transactions. RAP is triggered on quarterly basis and includes a variety of steps that link and cross-reference the Reactome content with many other resources (e.g UniProt, Ensembl etc), create inferences to other species, index the content, produce mapping and other intermediate files to later be used by the second group of tools in production and assist other editorial tasks. Data curated in human is finally inferred to other species through a series of steps during the release phase (<https://reactome.org/documentation/inferred-events>).

The second group includes the RESTful API (Application Program Interface), the Pathway Browser, the Pathway and Expression Analysis tools, and the ReactomeFIViz. Developed in JAVA and on top of the GKInstance library, the RESTful API provided programmatic, read only access to the Reactome content. The Pathway Browser, is a web application that relies on the RESTful API and enables users to access and visualise the Reactome content through a web browser without the burden of downloading and installing any additional software. This web application is developed in JAVA using the GWT framework.

The Pathway Analysis tool analyses user-supplied lists of genes, proteins and small molecules and provides ID mapping, pathway assignment and overrepresentation analysis. Both developed in JAVA, overrepresentation and expression analysis tools accept gene and protein accession numbers and identifiers that are associated with popular commercial platforms (e.g. Illumina, Agilent and Affymetrix). Analysis results can be both presented in a tabular form and visualised on top of pathways within the Pathway Browser.

The ReactomeFIViz [Wu et al. 2014] is a Cytoscape [Smoot et al. 2011] app designed to find pathways and network patterns related to cancer and other types of diseases. The app accesses the Reactome content, helps users perform pathway enrichment analysis for a set of genes, visualise hit pathways using manually laid-out pathway diagrams directly in Cytoscape, and investigate functional relationships among genes in hit pathways. ReactomeFIViz is developed in JAVA using the Cytoscape API.

Despite Reactome's rich software ecosystem, its tools and services were faced with a series of challenges revolving around concerns about functionality, performance and user friendliness that needed to be addressed.

1.3. AIM AND OBJECTIVES

The aim of this work is the study and implementation of engineering solutions to optimise the performance of biological pathway data storage and retrieval as well as improving its interactive visualisation. More specifically, the study and the development was done within Reactome scope and use cases, but the resulting techniques, data structures and strategies can be applied to a wider range of pathway database resources or other type of resources where the data model and/or needs are equivalent.

After evaluating the status of the tools and services provided by the resource and analysing the set of user requirements gathered by the Reactome team, the focus was set on the need of:

- Improving the existing version of Pathway Browser.
- Design and develop a better performing version of the pathway analysis algorithm and associated web service.
- Finding an easy way to graphically represent the pathways and their parent-child relationships to provide a means of overlaying analysis results so users can easily distinguish the most significant areas of biology represented in their data.
- Create a new version of the Diagram Viewer to improve the performance and enable it to deal with different views depending on the level of zoom.
- Change the underlying data storage mechanism to take advantage of the graph database technologies.

Although a number of subprojects of this thesis have been published separately in different journals, others needed to be dissertated in this document to bring a complete picture. The author then opted to create a chapter per each main topic enumerated above.

1.4. REFERENCES

Bader,G.D. Cary,M.P. Sander,C. (2006) Pathguide: a Pathway Resource List. *Nucleic Acids Research*, 34(suppl_1), D504-D506.

Cerami,E.G. Gross,B.E Demir,E. Rodchenkov,I Babur,Ö. Anwar.N. et al. (2011) Pathway Commons, a web resource for biological pathway data, *Nucleic Acids Research*, 39, D685-D690.

Croft,D. Fabregat,A. Haw,R. Milacic,M. Weiser,J. Wu,G. et al. (2013) The Reactome Pathway Knowledgebase. *Nucleic Acids Research*, 42, D472-D477.

Garcia-Campos,M.A. Espinal-Enriquez,J. Hernandez-Lemus,E. (2015) Pathway analysis: State of the art. *Frontiers in Physiology*, 6, 383.

Gawron,P. Ostaszewski,M. Satagopam,V. Gebel,S. Mazein,A. Kuzma,M. et al. (2016) MINERVA - a platform for visualisation and curation of molecular interaction networks. *Systems Biology and Applications*, 2, 16020.

Gehlenborg,N. O'Donoghue,S.I. Baliga,N.S. Goesmann,A. Hibbs,M.A. Kitano,H. et al. (2010) Visualization of omics data for systems biology. *Nature Methods*, 7, S56-S68.

GO Consortium (2017) Expansion of the Gene Ontology knowledgebase and resources, *Nucleic Acids Research*, 45(1), D331-D338.

Hastings,J. de Matos,P. Dekker,A. Ennis,M. Harsha,B. Kale,N. et al. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Research*, 41, D456-D463.

Kanehisa,M. Sato,Y. Kawashima,M. Furumichi,M. Tanabe,M. (2014) KEGG as a reference resource for gene and protein annotation, *Nucleic Acids Research*, 42(1), D199-D205.

Kanehisa,M. Furumichi,M. Tanabe,M. Sato,Y. Morishima,K. (2017) KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, Volume 45(1), D353-D361.

Karp,P.D. (2001) Pathway Databases: A Case Study in Computational Symbolic Theories. *Science*, 293(5537), 2040-2044.

Kuperstein,I. Cohen,D.P. Pook,S. Viara,E. Calzone,L. Barillot,E. et al. (2013) NaviCell: a web-based environment for navigation, curation and maintenance of large molecular interaction maps. *BMC Systems Biology*, 7, 100.

Kutmon,M. Riutta,A. Nunes,N. Hanspers,K. Willighagen,E.L. Bohler,A. et al. (2016) WikiPathways: capturing the full diversity of pathway knowledge. *Nucleic Acids Research*, 44, D488-D494.

Law,E, Roto,V. Hassenzahl,M. Vermeeren,A. Kort,J. (2009) Understanding, scoping and defining user experience: a survey approach. In: *Proceedings of the CHI 2009 Conference on Human Factors in Computing Systems*. ACM, New York, 19, 728.

Le Novère,N. Hucka,M. Mi,H. Moodie,S. Schreiber,F. Sorokin,A. et al. (2009) The Systems Biology Graphical Notation. *Nature Biotechnology*, 27, 735-741.

Lindgaard,G. Fernandes,G. Dudek,C. Brown,J. (2011) Attention web designers: You have 50 milliseconds to make a good first impression! *Behaviour & Information Technology*, 25(2), 115-126.

Marco-Ramell,A. Palau-Rodriguez,M. Alay,A. Tulipani,S. Urpi-Sarda,M. Sanchez-Pla,A. et al. (2018) Evaluation and comparison of bioinformatic tools for the enrichment analysis of metabolomics data. *BMC Bioinformatics*, 19, 1.

Moyano,T.C. Vidal,E.A. Contreras-López,O. Gutiérrez,R.A. (2015) Constructing simple biological networks for understanding complex high-throughput data in plants. *Methods in Molecular Biology*, 1284, 503-526.

- NCBI Consortium. (2018) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 46(1), D8-D13.
- Smoot,M.E, Ono,K, Ruscheinski,J, Wang,P.L. Ideker,T. (2011) Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27, 431-432.
- Uniprot Consortium. (2017) UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(1), D158-D169.
- Vastrik,I. D'Eustachio,P. Schmidt,E. Joshi-Tope,G. Gopinath,G. Croft,D. et al. (2007) Reactome: a knowledge base of biologic pathways and processes, *Genome Biology*, 8, R39.
- Wu,G. Dawson,E. Duong,A. Haw,R. Stein,L. (2014) ReactomeFIViz: a Cytoscape app for pathway and network-based data analysis. *F1000Research*, 3, 146.
- Zerbino,D.R. Achuthan,P. Akanni,W. Amode,M.R. Barrell,D. Bhai,J. et al. (2018) Ensembl 2018. *Nucleic Acids Research*, 46(1), D754-D761.

2. PATHWAY BROWSER

The Pathway Browser (<https://reactome.org/PathwayBrowser/>) is Reactome's primary means of viewing and interacting with pathways data. Due to the inherent complexity of the represented knowledge, creating a long-lasting large-scale tool to interactively browse Reactome content presents a set of significant challenges to overcome. These challenges comprise specific aspects such as choosing the programming language or framework, as well as more generic concepts such as the code layout and data retrieval strategies. These were addressed with the objective of offering a product that fulfils the user's requirements.

User requirements and needs typically evolve over time requiring software engineers to design applications in a way that reduces the hurdle of adding new features or extra capabilities as much as possible. This is a very well know problem in computer science and there are several established techniques to help developers minimise the impact of changes or additions.

This chapter begins with a description of the status of the Pathway Browser prior to the development work undertaken for this thesis and later explains the requirements and options considered for the re-engineering, focusing on four main aspects: (i) the code layout pattern, (ii) the strategy followed to improve the memory consumption, (iii) a discussion of the State Manager module and finally (iv) the different widgets (either reused from third parties or developed) to display data from other resources within the Pathway Browser.

2.1. REACTOME PATHWAY BROWSER STATE OF THE ART

In its early years, Reactome relied on a set of dynamically-generated web pages to enable users to browse its content. The Event Browser was an aggregation of pages generated using Perl-CGI scripts to present a basic User Interface (UI) and limited means for user interaction (Figure 2.1) [Vastrik et al. 2007].

Released in 2011, the first version of the Pathway Browser was developed on top of the Event Browser, aiming to support pathway analysis and visual navigation based upon the Systems Biology Graphical Notation (SBGN) [Croft et al. 2010]. To address the needs of this first version, two new features were added in the event browser; a Hierarchy Tree and a pathway Diagram Viewer (Figure 2.2). The application was developed in JavaScript, employing an early version of the Yahoo User Interface (YUI) framework [Croft et al. 2010].

Although this application provided an improved way to browse Reactome content, it was faced with a set of major challenges that significantly impeded further development. Originally developed to be a small-scale tool, the application grew organically with new features being added upon request. Since the code structure did not follow a modular approach, it soon resulted in a set of large and difficult to manipulate files with no clear segregation of responsibilities. This made the addition of new features a challenging and inefficient task.



Figure 2.1. Event Browser. Reactome’s early tool for accessing its content.

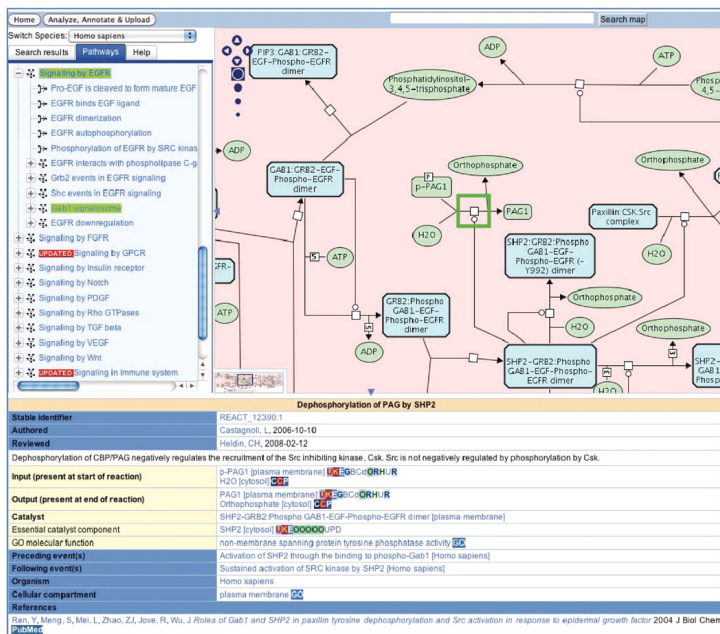


Figure 2.2. The first version of the Reactome Pathway Browser.

2.2. RE-ENGINEERING THE PATHWAY BROWSER

As Reactome content increased, the limitations of the existing tool brought it to an unsustainable point where a re-engineering phase became necessary to cope with fast growing users' demands. From the beginning of the redesign, one of the main requirements was that the new Pathway Browser had to be implemented in a modular way and thus allow for long term maintenance. Developing any large-scale application, such as the Pathway Browser, has its hurdles. On one hand, implementing a custom solution enables full control over features and capabilities, though at the cost of longer development time. On the other hand, reusing existing software has the advantage of launching the final product in a shorter period of time but opportunities to add features are limited by the capabilities of the existing third-party software [Krueger 1992].

The new Pathway Browser had to incorporate Reactome's pathway and expression Analysis Tools (Chapter 3). These tools enable mapping of entity identifiers and gene symbols to pathways, over-representation analysis and expression data overlay of user data sets. Additionally, since Reactome includes pathway annotation and supports analysis in other species (<https://reactome.org/documentation/inferred-events>), the new version had to feature a simple species-switching mechanism.

Like many other resources, Reactome tools have always been subject to the limitations of available web technologies. Reactome's website has evolved over the years as (i) new technologies provided more advanced tools, (ii) faster and more robust web browsers were released and (iii) better web technologies were implemented. The rise/expansion of HTML5, AJAX and HTML5 Canvas were the cornerstone of modern web development. An

additional problem that Reactome has needed to address is the problem of cross-browser compatibility and differences between versions of the same browser. Therefore, while retaining the main features of its predecessor but taking into account the requirements discussed above and considering the need for a long-term support strategy, a new version of the Pathway Browser was implemented from scratch as an aggregation of components and modules using GWT Toolkit (<http://www.gwtproject.org/>). GWT is a development toolkit for building and optimizing complex browser-based applications. Its goal is to enable productive development of high-performance web applications. It allows writing client-side applications in Java and then compile the source to highly optimised JavaScript that runs across all browsers. The GWT compiler performs comprehensive optimisations across the codebase such as in-lining methods, removing dead code or optimizing strings.

This section is divided into five subsections that cover different aspects of the new implementation, namely (i) the software layout pattern, (ii) ways to improve memory consumption, (iii) strategies to control the status of the application, (iv) how different widgets were developed or reused from third parties to include data from other resources, and (v) a subsection describing the resulting web application.

2.2.1. SOFTWARE LAYOUT PATTERN. MODEL VIEW PRESENTER (MVP)

Whatever the selected programming language and frameworks, as previously mentioned, developing any large-scale application comes with a set of challenges. Perhaps the most important of these when developing a large-scale application is establishing a clean code base with a clearly defined architecture that can be easily maintained and extended in the future.

To accomplish this, the separation of concerns (SoC) principle was used to design the new architecture. This dictates that computer programs should be separated into distinct sections/modules, such that each section addresses a separate concern. SoC can be achieved by encapsulating information inside a module that has a well-defined interface. By following this approach, individual modules can be reused, as well as developed and updated independently. In addition, a module can be modified without having to know the details of other modules, and without having to introduce corresponding changes to those modules, thus, simplifying development and maintenance of computer programs.

Adopting a layered architecture becomes even more important in applications that feature a complex user interface (UI). Over the past years, a number of design patterns have been introduced aiming to compartmentalise areas of responsibility within an application. Design patterns such as Presentation-Abstraction-Control (PAC), Model-View-Controller (MVC), or Model-View-Presenter (MVP), come with distinct benefits, all having the potential to decrease application development time to focus on different modules of the application.

The MVP architecture was selected for the new version of the pathway browser. Figure 2.3, illustrates the position and role of each of the three main actors in the pattern. In particular: (i) the Model encompasses business objects, (ii) the View contains all of the UI components of the application, including any tables, labels, buttons, text boxes, etc. The View is also responsible for the layout of the UI components. It is worth mentioning that in MVP the View does not have any direct interaction with the Model, (iii) the Presenter contains all of the business logic for the application and acts as the mediator between Model and View. As a general rule, for every view there

is a presenter to drive it and handle events that are sourced from the UI widgets within the view.

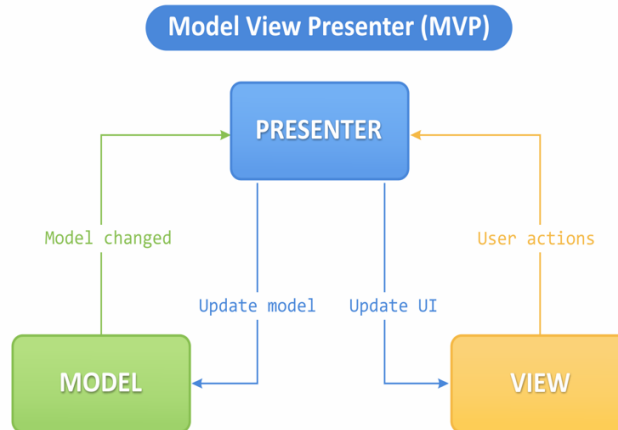


Figure 2.3. The position and role of each of the three main actors in the MVP pattern.

Although very similar to MVC, MVP has a fundamental difference; it replaces the Controller with a Presenter. The latter can update the View directly, something that is not allowed for a Controller. Also, the Presenter is responsible for handling all user events raised from the View. As a result, all the complex view logic can be removed from the View and included in the Presenter, leaving the View with only the drawing logic. Therefore, the MVP architecture achieves a finer and clearer separation of functionality between business logic and UI, decoupling development in a way that allows multiple developers to work simultaneously. Additionally, adopting MVP makes automated unit testing simpler as the Presenter can be tested without direct need of screen elements, by creating a test class implementing the interface of the View.

As previously explained, the Pathway Browser follows a modular architecture. All components comprising the Pathway Browser can be classified into two main categories: (i) Modules are components having to update their UI elements and thus featuring their own View, e.g. the Hierarchy Panel, or the Pathway Diagram Viewer and (ii) managers are components without UI, which do not have a View, e.g. the State Manager.

2.2.2. EVENT BUS

Communication among the different modules and managers within the Pathway Browser is achieved via the use of an Event Bus (Figure 2.4). The latter is a publish/subscribe-style mechanism for dispatching events to interested parties, without requiring the components to explicitly be aware of each other. Its major advantage is that it eases decoupling by allowing components to interact without having direct dependencies upon one another, and without requiring event sources to deal with maintaining handler lists.

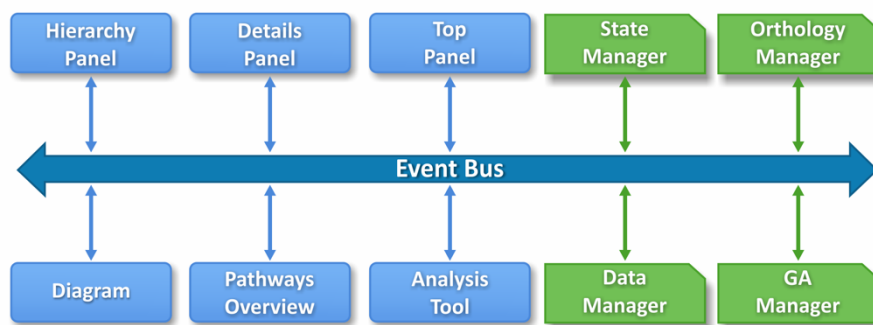


Figure 2.4. Event bus with the different modules and managers.

The use of an Event Bus can assist in producing scalable applications with loosely coupled modules. However, it should not be overused, in particular, not all events should be placed on the Event Bus, as this could lead to ‘chatty’

applications that get slowed down by event handling. In other words, event proliferation should be avoided as it results in a fair amount of boilerplate code to define, source, sink, and act upon these events.

2.2.3. STATE MANAGER

When updating the content of a page, making multiple requests to the web server for mark-up is unnecessary. JavaScript is capable of loading content, updating parts or displaying and hiding fragments of the Document Object Model (DOM) without reloading the whole page. Asynchronous JavaScript And XML (AJAX) uses a combination of browser built-in XMLHttpRequests and JavaScript to allow web pages to be updated asynchronously, by exchanging data with a web server behind the scenes. As a result, AJAX has become the main data pipeline, with requests occurring in the background.

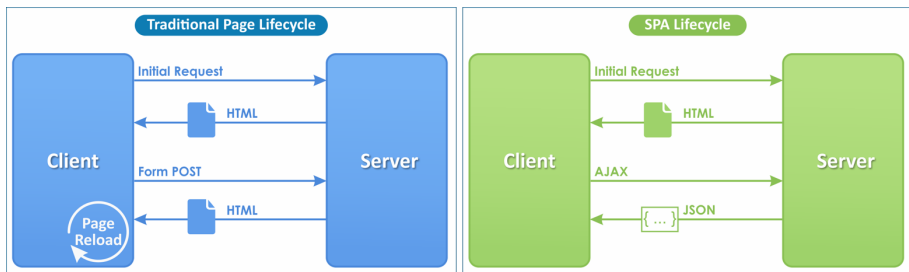


Figure 2.5. The traditional page life cycle versus the SPA life cycle (adapted from <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>).

This model of web application that loads a single HTML page and dynamically updates that page as the user interacts with the application using AJAX and HTML5, without constant page reloads, is commonly referred to as a Single Page Application (SPA). Figure 2.5 illustrates how the SPA lifecycle compares to the traditional one. The SPA model has become very

important since it has contributed to transforming web applications into their native desktop application counterparts in terms of look and feel.

Developers have gradually adopted the SPA model as it provides a richer experience for the final users. However, abandoning the traditional page lifecycle in favour of performing asynchronous background requests and handling page rendering has one serious side effect with significant usability implications; the browser's history managing mechanism gets systematically circumvented, rendering the browser back and forward buttons useless [Mikowski and Powell 2013].

Since the Pathway Browser was intended to run inside a web browser as an SPA, one of the main challenges posed was to maintain its state and allow users to take advantage of existing history management features. Simply put, the user should be able to use the web browser's back and forward buttons to seamlessly control the state of the application.

This is why developing the State Manager within the Pathway Browser became vital. This manager, also referred to as the History Manager, is responsible for keeping track of any changes in the state of the application and passively update the URL to notify the web browser of such a change. To accomplish this, the module (i) subscribes to the event bus (Figure 2.4), listens for relevant events, as a result of the user interaction, and updates accordingly the URL on the browser's address bar and (ii) listens for any change in the address bar and fires an event in the event bus that notifies the rest of the modules about the change in state.

The State Manager changes only one parameter of the URL at a time based on user actions and by doing so, it enables the browser back and forward button capability. Since the application keeps track of the state for each

interaction and updates the URL, a given state or view in the Pathway Browser can be shared by simply sharing the URL. As a result of opening the shared URL, the recipient will be placed in the exact same view thanks to the State Manager.

Technically speaking, the State Manager can be seen as an approximation to a finite state machine model, which is a well-studied model for describing a synchronous sequential machine [Moore 1964]. Its implementation is as simple as powerful and is one of the key features of the new version of the Pathway Browser together with the data retrieval strategy to keep a low memory consumption allowing users to use the web application for longer periods of time without losing performance.

2.2.4. MEMORY CONSUMPTION

The advent of the Web 2.0 era, a term introduced back in 1999 [DiNucci 1999], gave birth to a new breed of web applications that emphasised user-generated content, usability and interoperability for end users. Modern web applications provide more functionality, richer user experience and improved performance. Nevertheless, all these benefits came at the cost of increased complexity.

Nowadays, web applications have become more and more client-side oriented and it is very common for a user to stay on a single web page for hours without leaving. The web browser just retrieves data from the server through asynchronous requests and displays it in ever more interesting ways. As a result, over the past years, web applications have become greedy for resources, at a pace that browsers cannot keep up with. The intense utilisation of JavaScript to interact with the Document Object Model (DOM) exposed the poor memory management of most major browsers of the time. The

Pathway Browser is an example for this kind of application, so properly managing the memory consumption was an issue to be taken into account.

Although web browsers have evolved, improving their memory management, most of these issues persist. Developers should not only be aware of them, but they should also prevent them in their applications. However, there seems to be some confusion when it comes to separating memory leak issues from memory consumption.

Memory consumption stands for the amount of memory a particular program requires throughout its execution. Since every application relies on the underlying memory to store its variables and data structures, it is a common assumption that more complicated applications require more memory. A memory leak is a specific case of misusing the memory. Memory leaks are bugs in the memory management of either the web browser or the application itself. A leak refers to a situation in which a program occupies a piece of memory that later it does not release, resulting in this memory piece becoming useless and inaccessible. Depending on its size, a single leak can be harmless. However, by repeating this process over time, leaks can accumulate leading to paging and eventually memory starvation. Simply put, the application becomes slower, less responsive and eventually crashes as it is unable to continue the execution due to insufficient memory resources.

As mentioned before, modern web browsers have improved their memory management over the years. Nevertheless, since memory leaks are mainly caused by specific programming patterns, almost any browser can leak memory under certain circumstances. Thus, understanding what causes them is valuable. Leaks can be classified in the following categories according to Justin Rogers under “Understanding and Solving Internet Explorer Leak

Patterns” in the msdn documentation (<http://msdn.microsoft.com/en-us/library/Bb250448>):

- Circular References are the most common cause of leaks. Objects can leak memory when mutual references are counted between the browser’s infrastructure and any scripting engine.
- Closures are often responsible for leaks because they create circular references, as parent function parameters and local variables will be frozen in time, referenced, and held until the closure itself is released.
- Cross-Page Leaks refer to very small leaks of internal bookkeeping objects while moving from site to site.
- Pseudo-Leaks. Although these cannot be considered leaks, it refers to extremely annoying scenarios when the over usage of memory cannot be understood.

Aiming to minimise memory consumption and prevent any memory leaks that would compromise stability and performance, a well-engineered data retrieval strategy was adopted in the new Pathway Browser. To avoid excessive memory usage objects were loaded on demand and cached before their re-use to avoid unnecessary requests to the server. Moreover, the objects are stored in a Least Recently Used (LRU) list, so when the list is full the new objects will replace those that have not been used for longer. To address DOM memory leaks across all modern browsers, development was based on the GWT Toolkit. The latter deals with each browser’s peculiarities and ensures efficient garbage collection. Also, special attention was given to every instantiation of a new object and disposal of those that were no longer required.

2.2.5. WIDGETS FOR THIRD PARTY DATA INTEGRATION

The main modules in the Pathway Browser were designed to represent Reactome content. However, there are circumstances when integrating data from other resources might help the user gain a better understanding of the represented biology. To enrich the Pathway Browser with such a capability, third party widgets were used where available, and a number of modules were newly developed as widgets. These retrieve data from third party resources and show their content based on items selected in the Pathway Browser.

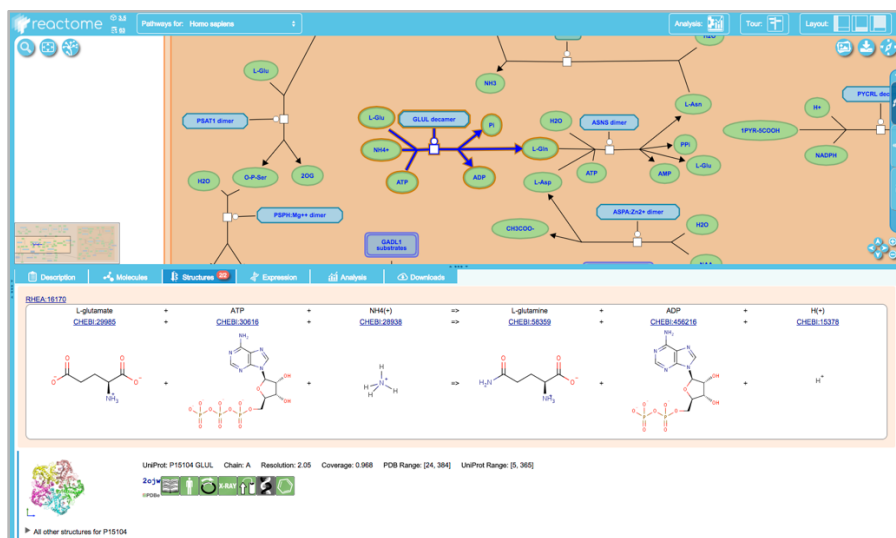


Figure 2.6. Pathway Browser main viewport and “Structures” tab. The main viewport is showing part of the pathway “Amino acid synthesis and interconversion (transamination)” with the selected reaction “Glutamate + NH₄⁺ + ATP => glutamine + ADP + orthophosphate [GLUL]”. “Structures” is the active tab in the Details Panel and it shows two widgets. The widget on the top shows data from Rhea and the one below from PDBe. (<https://reactome.org/PathwayBrowser/#/R-HSA-70614&SEL=R-HSA-70606&DTAB=ST>).

Table 1.1. Entity type with its associated resource and its description. Data from these resources are retrieved and shown by the widgets shown in the “Structures” tab within the Details Panel

Entity type	Resource	Resource Description
Compound	ChEBI https://www.ebi.ac.uk/chebi/ [Hastings et al. 2013]	Freely available dictionary of molecular entities focused on ‘small’ chemical compounds. The term ‘molecular entity’ refers to any constitutionally or isotopically distinct atom, molecule, ion, ion pair, radical, radical ion, complex, conformer, etc., identifiable as a separately distinguishable entity. The molecular entities in question are either products of nature or synthetic products used to intervene in the processes of living organisms.
Protein Structure	PDBE https://www.ebi.ac.uk/pdbe [Mir et al. 2018]	Integrated resource of high-quality macromolecular structures and related data.
Reaction	Rhea http://www.rhea-db.org/ [Morgat et al. 2017]	Expert curated resource of biochemical reactions designed for the annotation of enzymes and genome-scale metabolic networks and models.

These widgets are integrated in two different tabs of the Details Panel; (i) the “Structures” tab and (ii) the “Expression” tab. The “Structures” tab shows a set of widgets from a given resource depending on whether a reaction, protein or compound is selected (Figure 2.6). When a complex or set is selected, their participants are retrieved and listed in the “Structures” tab and the appropriate resource widget is used for each of them. Table 1.1 shows the entity types and resources from which data are retrieved.



Figure 2.7. Pathway Browser main viewport and “Expression” tab. The main viewport is showing part of the pathway “Amino acid synthesis and interconversion (transamination)”. Expression is the active tab in the Details Panel and it shows the stand-alone widget provided by Expression Atlas (<https://reactome.org/PathwayBrowser/#/R-HSA-70614&DTAB=EX>).

In the “Expression” tab, one of the several stand-alone widgets available from Expression Atlas (<https://www.ebi.ac.uk/gxa>) was integrated (Figure 2.7). Expression Atlas is an open science resource that gives users a powerful way to find information about gene and protein expression across species and biological conditions such as different tissues, cell types, developmental stages and diseases, among others. The widget receives a set of identifiers and shows condition specific gene expression data that is stored in this resource.

Main web browsers implement the Same-Origin Policy (SOP) which prevents scripts from accessing data from servers other than that which originally served the page. SOP is a security measure enforced by browsers that restricts

malicious websites from running JavaScript inside external websites. In many cases, widgets developed by a given resource R1 to be used in third party resources need to access data that are hosted in the R1 server which is not the server that will provide the main content of the page where the widget will be hosted.

There are two ways to avoid the SOP: (i) using a proxy in the third party resource server side to redirect calls to the server hosting the targeted data or (ii) making use of Cross-Origin Resource Sharing (CORS). CORS is a mechanism that uses additional HTTP headers to let a user agent gain permission to access selected resources from a server on a different origin (domain) than the site currently in use. These two techniques were used in both the implemented and the reused widgets for third party data integration; ChEBI and Rhea widgets were implemented to use the proxy approach whilst the remaining widgets take advantage of CORS.

2.2.6. RESULTING WEB APPLICATION

The reengineering of the Pathway Browser described in this thesis has taken place over a period of several years. The application has gradually improved in both design and performance [Croft et al. 2013; Fabregat et al. 2015 and 2017].

The screenshot shows the Reactome Pathway Browser interface. On the left is the 'Hierarchy Panel' with a tree of biological events. The main area is the 'Viewport', which is split into 'Pathways Overview' and 'Diagram Viewer'. The 'Details Panel' at the bottom shows the 'Summation' of the pathway, describing the synthesis of inosine 5'-monophosphate (IMP) from 5-phospho-alpha-D-ribose 1-diphosphate (PRPP).

Summation

The purine ribonucleotide inosine 5'-monophosphate (IMP) is assembled on 5-phospho-alpha-D-ribose 1-diphosphate (PRPP), with atoms derived from aspartate, glutamine, glycine, N10-formyl-tetrahydrofolate, and carbon dioxide. Although several of the individual reactions in this sequence are reversible, as indicated by the double-headed arrows in the diagram, other irreversible steps drive the pathway in the direction of IMP synthesis in the normal cell. All of these reactions are thus annotated here only in the direction of IMP synthesis. Guanosine 5'-monophosphate (GMP) and adenosine 5'-monophosphate (AMP) are synthesized from IMP (Zaklin & Dixon 1992).

The pyrimidine uracil (uracil acid) is synthesized in a sequence of four reactions, deriving its atoms from glutamine, bicarbonate, and aspartate. A single multifunctional cytosolic enzyme catalyzes the first three of these reactions, while the last one is catalyzed by an enzyme associated with the inner mitochondrial membrane. In two further reactions, catalyzed by a bifunctional cytosolic enzyme, uracil reacts with 1-phosphoribosyl 5-phosphosphate (PRPP) to yield uridine 5'-monophosphate, which is decarboxylated to yield uridine 5'-monophosphate (UMP). While several individual reactions in this pathway are reversible, other irreversible reactions drive the pathway in the direction of UMP biosynthesis in the normal cell. All reactions are thus annotated here only in the forward direction.

This pathway has been most extensively analyzed at the genetic and biochemical level in hamster cell lines. All three enzymes have also been purified from human sources, however, and the key features of these reactions have been confirmed from studies of this human material (Jones 1985).

All other pyrimidines are synthesized from UMP. The reactions annotated here, catalyzed by dCMP deaminase and dUTP diphosphatase yield dUMP, which in turn is converted to TMP by thymidylate synthase.

Figure 2.8. Pathway Browser illustration highlighting the different panels. Access to events is provided via the ‘hierarchy panel’ of events on the left and by clicking on event nodes in the Pathways Overview within the ‘viewport’. More information for the selected entity is shown in the ‘Details Panel’. When a pathway is opened, the Diagram Viewer is revealed in the ‘viewport’ replacing the Pathways Overview. Buttons to the right of the logo in the ‘header panel’ show the current software version (3.5) with access to the Github software repository, and the current version of Reactome data (release 63). A button to the right of the ‘header panel’ provides access to the Analysis Tools. Clicking on the layout buttons (top right) closes and reopens the hierarchical display and Details Panel. The ‘tour’ button provides access to a brief video tour of the main features.

Amongst the improvements, the Pathway Browser (Figure 2.8) was re-engineered to reduce loading time and provide a more attractive user interface. Buttons for widely used actions were made more prominent, icons and colour schemes were re-designed and features including colour profiles were added to allow user customisation. The Pathway Browser now opens with the

Pathways Overview (Chapter 4) visible in the main viewport. This overview is integrated in the Pathway Browser main view, sharing the location used by the Diagram Viewer (Chapter 5), used to display pathway diagrams representing detailed molecular events. When the Pathway Browser loads, the Events Hierarchy and the Details Panel appear on the left and bottom of the viewport, respectively. Double-clicking a pathway in the events hierarchy or its node in the Pathways Overview triggers a zoom animation in the main Viewport leading to the equivalent pathway diagram.

The components of the Pathway Browser are connected through the Event Bus, so that user actions affecting the display of one component will cause other components to update, presenting information consistently across the display elements. For example, selecting a reaction node or a physical entity glyph in the pathway diagram will trigger an update of the information displayed in the Details Panel, located below the pathway Diagram Viewer, and in the Events Hierarchy panel on the left side.

2.3. SUMMARY

Reactome software has evolved over the years, aiming to provide bioinformatics tools for visualisation, interpretation and analysis of pathway knowledge to support basic research, genome analysis, modelling, systems biology and education. Among the tools, the Pathway Browser has become Reactome's main means for accessing its content. It has to be easy to use and responsive in order to offer a good user experience.

Creating a long-lasting large-scale tool presented a set of significant challenges to overcome. The new version of Pathway Browser was developed using the MVP software layout pattern to (i) decrease the application

development time, (ii) separate the view implementation from the business logic and (iii) to provide with a scalable and easy to maintain tool.

Controlling the memory consumption and preventing memory leaks were other challenges to be addressed. To avoid excessive memory usage, objects are loaded on demand and cached in an LRU list. To control DOM memory leaks across all modern browsers, the development was based on the GWT Toolkit, so the framework itself that takes care of this.

A State Manager was put in place to maintain the internal application state and allow users to take advantage of the existing browser history management features. The benefit of it is to allow users to use the web browser's back and forward buttons to seamlessly control the state of the application.

All the components and modules in the Pathway Browser are tightly connected through the Event Bus, so that user actions affecting the display of one component will cause other components to update, presenting information consistently across the display elements.

Integrating data in the Pathway Browser from other resources helps users to gain a better understanding of the represented biology. To accomplish this task, third party widgets were used, when available and a number of modules were newly developed as widgets. These retrieve data from third party resources and show their content based on items selected in the Pathway Browser.

The Pathway Browser was re-engineered to reduce loading time and provide a more attractive user interface. It also integrates the Analysis Tools, allowing users to submit their data and then navigate through the results taking advantage of the other integrated modules.

2.4. REFERENCES

Croft,D. Fabregat,A. Haw,R. Milacic,M. Weiser,J. Wu,G. et al. (2013) The Reactome Pathway Knowledgebase. *Nucleic Acids Research*, 42, D472-D477.

Croft,D. O’Kelly,G. Wu,G. Haw,R. Gillespie,M. Matthews,L. et al. (2010) Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39, D691-D697.

DiNucci,D. (1999) *Fragmented Future*. Print, 53(4), 221-222.

Fabregat,A. Jupe,S. Matthews,L. Sidiropoulos,K. Gillespie,M. Garapati,P. et al. (2017) The Reactome Pathway Knowledgebase. *Nucleic Acids Research*, 46, D649-D655.

Fabregat,A. Sidiropoulos,K. Garapati,P. Gillespie,M. Hausmann,K. Haw,R. et al. (2015) The Reactome pathway Knowledgebase. *Nucleic Acids Research*, 44, D481-D487.

Hastings,J. de Matos,P. Dekker,A. Ennis,M. Harsha,B. Kale,N. et al. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Research*, 41, D456-D463.

Krueger,C.W. (1992) Software reuse. *Computing Surveys*, 24(2), 131-183.

Mikowski,M. and Powell,J. (2013) *Single Page Web Applications: JavaScript end-to-end*, Manning Publications Co., Greenwich, Connecticut, USA.

Mir,S. Alhroub,Y. Anyango,S. Armstrong,D.R. Berrisford,J.M. Clark,A.R. et al. (2018) PDBe: towards reusable data delivery infrastructure at protein data bank in Europe. *Nucleic Acids Research*, 46, D486-D492.

Moore,E.F. (1964) *Sequential Machines Selected Papers*. Wesley series in computer science and information processing. Addison Wesley Publishing Company Inc., Reading, Mass.

Morgat,A. Lombardot,T. Axelsen,K.B. Aimo,L. Niknejad,A. Hyka-Nouspikel,N. et al. (2017) Updates in Rhea – an expert curated resource of biochemical reactions. *Nucleic Acids Research*, 45, D415-D418.

Vastrik,I. D'Eustachio,P. Schmidt,E. Joshi-Toppe,G. Gopinath,G. Croft,D. et al. (2007) Reactome: a knowledge base of biologic pathways and processes, *Genome Biology*, 8, R39.

3. PATHWAY ANALYSIS TOOLS

In bioinformatics research, pathway analysis software is used to compare a set of genes or proteins, usually generated by an ‘omics method, to canonical prior knowledge structured in the form of pathways. Pathway analysis methods have a broad range of applications in physiological and biomedical research. These methods help researchers discover which areas of biology, and biomolecules, are crucial to understand the phenomena under study. However, pathway analysis methods should never be taken as black boxes where experimental data goes in and true statements come out, rather they are better considered as metal detectors, helping researchers to find biologically meaningful needles in the proverbial haystack [García-Campos et al. 2015].

Reactome's annotated pathway data represents the molecular details of events that are sufficiently well-established to be agreed upon by experts in the area. They show what could happen if all annotated proteins and small molecules were present simultaneously in a generic human cell. By overlaying an experimental dataset on these annotations, such as a list of proteins activated in response to an experimental stimulus, or genes expressed in transformed cells but not their normal counterparts, a user can identify modulation of specific pathways. By overlaying quantitative expression data, such as a time series or stages of disease development, a user can visualise the extent of change and its progression in affected pathways.

The first section of this chapter discusses the previous approach using a relational database to support the pathway analysis. The second section summarises the new approach described in the paper “Reactome pathway analysis: a high-performance in-memory approach” [Fabregat et al. 2017] (Publication P.1) and presents the new tools built on top of the latter to cope

with Reactome users needs such as the analysis service and its lightweight client, which is integrated in the Pathway Browser.

3.1. THE RELATIONAL DATABASES APPROACH

Relational databases are widely used in pathway databases for data management; either during curation, the release process or in the final production phase. It is also very common to store the information in third normal form due to its convenience for data integrity assurance [Chowdhury 2015; Shin and Sanders 2006; Codd 1972].

Relational databases in their third normal form can be efficient in computational terms. For the above-mentioned use cases, however, this approach greatly slows the execution of analysis algorithms, due to the size of the temporary tables for the queries and later projections. For this reason, database-based analysis approaches use denormalised versions of the databases instead [Talbi and Zomaya 2008]. The denormalisation process replicates a lot of data to speed up the queries but it may penalise analysis execution time as the original database content grows bigger.

Focusing on the computational side of the problem, the query containment problem is undecidable for relational algebra and SQL, but is decidable and NP-complete for conjunctive queries. In fact, the query containment problem for conjunctive queries is exactly the same problem as the query evaluation problem [Abiteboul et al. 1995]. When queries tend to be small, NP-completeness is usually considered acceptable but its performance falls when queries tend to be big. In addition, it is also worth considering that creating intermediate tables in memory after executing a “join” statement is one of the heaviest operations for a database engine.

Reactome's previous implementation of the pathway analysis was based on a denormalised version of the Reactome relational database. Among its limitations were that it provided results only of the higher-level pathways in Reactome, and the lack of programmatic access. In addition, the previous implementation suffered from poor performance mainly due to the fact that, on every analysis request, it connected to the relational database, rather than querying an intermediate in-memory data structure. Thus, the response time of the previous Reactome analysis could reach 5 min, as soon as the user sample included a few hundreds of gene identifiers, causing a high server load that, combined with a number of concurrent analysis requests, affected the stability of the Reactome website and often resulted in outages.

3.2. A HIGH-PERFORMANCE IN-MEMORY APPROACH

The use of high-throughput platform technologies has transformed biological research. High-throughput results often contain thousands of identifiers and the size of sample data is predicted to continue increasing [Reuter et al. 2015]. Growth in query size, increasing usage of analysis tools and expanding content all contribute to rapidly increasing demands on Reactome's Pathway Analysis Service. To address this, a new set of improved pathway Analysis Tools that can handle the increasing demands were developed to provide reliable and accurate results with interactive response times measured in seconds.

To better address the challenges mentioned in Section 3.1, a novel implementation of the enrichment analysis (ORA) method [Drăghici et al. 2003] was developed, focussing on the computer science aspect as well as elaborating on the different data structures and design patterns used to optimise execution time and reduce server load. This new implementation achieves interactive speed that is more than adequate for genome scale

datasets, typically providing results for a dataset of 5,000 identifiers in under 3 seconds. This new implementation also offers fine-grained results at all levels of Reactome's hierarchy of events. The analysis provides measures of target pathway coverage in terms of matching molecules and in terms of matching reaction-like events.

To solve the described NP-complete problem for the relational database approach, the procedure was to break down the analysis problem into subproblems that are simple enough to be solved in polynomial time by identifying a convenient data structure, the so-called 'divide and conquer' rule. The pathway overrepresentation analysis algorithm can be split into four parts: (i) Check whether the user's protein/chemical identifiers are present in Reactome, (ii) if present, determine whether they are part of complexes and/or sets as well as the species projection (Section 1.2), (iii) aggregate the found identifiers by the pathways (and super-pathways) where they occur and finally (iv) calculate the likelihood that the overlap between the identifiers and the pathway is due to chance.

In the first step, the main requirement is to efficiently determine whether identifiers in the sample correspond to one or many entities in Reactome (gene names for example can map to more than one protein identifier). This is best achieved by the reverse approach whereby a lookup table is created containing all the identifiers that correspond to entities in Reactome. This lookup table was constructed as a radix tree, (Figure 3.1a), which is a space-optimised trie data structure where nodes with only one child are merged with their parents [Morrison 1968]. A trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings [De la Briandais et al. 1959].

The data structure selected for the second part of the analysis had to model the entities composition problem, namely projection to other species orthologs and identifier to entity mapping. To achieve this a directed graph was utilised (Figure 3.1b). A directed graph is a graph, or set of nodes connected by edges, where the edges have a direction associated with them. For a given graph G with several nodes (a , b and c), if G has an arrow from a to b and another arrow from b to c , then the composed graph G_2 has an arrow from a to c .

The data structure used to model the third and fourth parts of the analysis is a double-linked tree (Figure 3.1c). In this structure, each node represents a pathway and contains links to its parent and children. When a node in the tree is hit, the action is recursively propagated all the way up to the root. To reduce the memory footprint, only identifiers, names and placeholders for results calculation are stored in the nodes. When the identifier search finishes, a binomial test is used to calculate the probability for each pathway node. The P-values are corrected for the multiple testing Benjamini–Hochberg procedure that arises from evaluating the submitted list of identifiers against every pathway [Benjamini and Hochberg 1995].

Summarising the steps (Figure 3.1), for each identifier in the user's sample, the first action is to find whether it is present in Reactome using a previously built radix-tree as a lookup-table. This significantly speeds up the process but requires a low memory footprint. For identifiers that are matched, the corresponding radix-tree nodes point to one or many other nodes in a graph which is used as the second data structure to store the curated relationships between Physical Entities and their other-species orthology. Traversing this second data structure, while applying or not the projection to species, provides pointers to all pathways stored in the final data structure, a double-linked tree,

which serves to aggregate the result and acts as a placeholder for the last step when the analysis statistics are calculated.

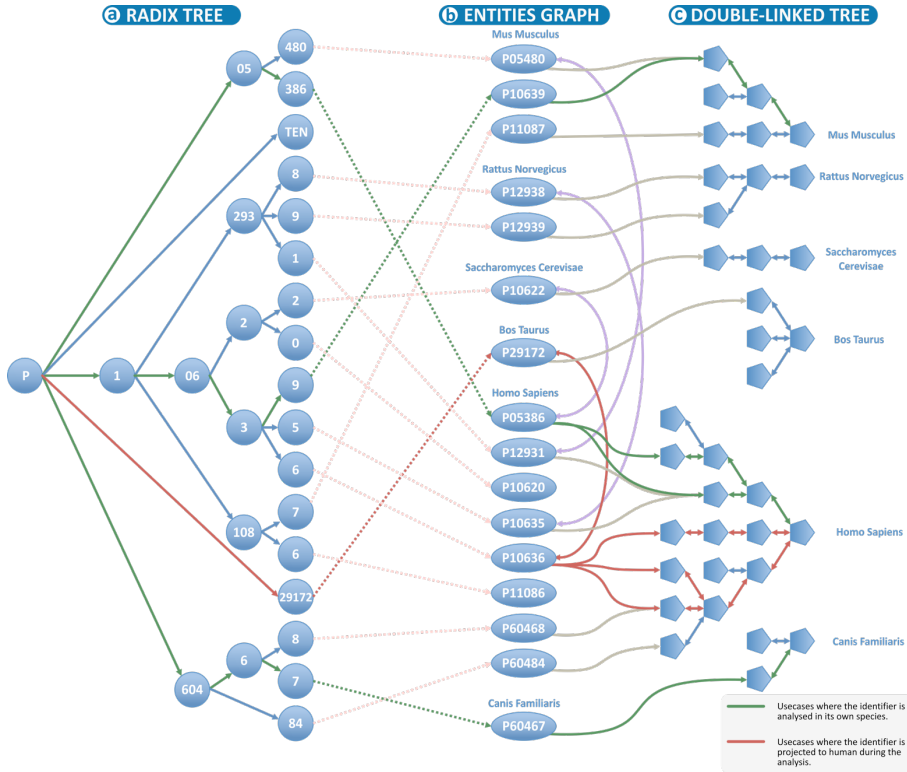


Figure 3.1. Representation of two analysis use cases joining the different data structures. In red an analysis performed using the projection to human. In green an analysis performed without projection [Fabregat et al. 2017].

Even though the presented analysis strategy processes each identifier in the submitted sample in a sequential fashion, different analyses requested at the same time are executed concurrently by separate threads at the Analysis Service level (Section 3.2.1). Finer grain parallelisation could be implemented in the future, if required, to further reduce the analysis time per each individual

request, at the cost of additional code complexity and therefore its associated maintenance burden.

3.2.1. NEW SOFTWARE ECOSYSTEM

Resources provide web services to facilitate their integration into third-party pipelines, scripts or applications. Once the analysis core algorithm was built, the next step was to provide a mechanism to make it accessible to the community. As an in-memory approach was used to achieve a fast-resolving algorithm, the populated data structures must be placed in memory while the service is running.

The task of populating the defined data structures with Reactome database content is relatively large, taking approximately 20 minutes to complete on a standard laptop featuring an Intel Core i7 at 2.6 GHz, 16 GB of DDR3 memory at 1,600 MHz, and 256 GB of flash storage. This is too slow for a usable service, so a strategy was put in place to make the data available when running the service to minimise the start time. To achieve this, an intermediate data file containing a serialised version of the data structures in memory was created, using Kryo (<https://github.com/EsotericSoftware/kryo>). Kryo is a fast, efficient object graph serialisation framework for Java. After running the analysis core builder, data structures placed in memory are serialised into an intermediate data file and stored on the hard drive (Figure 3.2a).

When the RESTful service for analysis is started, it reads from the intermediate data file (Figure 3.2b) and loads its content into memory in approximately one minute. Once content is loaded, the service is ready to serve user analysis requests. As one use-case for the Analysis Service, a lightweight client (Figure 3.2c) was developed and integrated into the Pathway Browser.

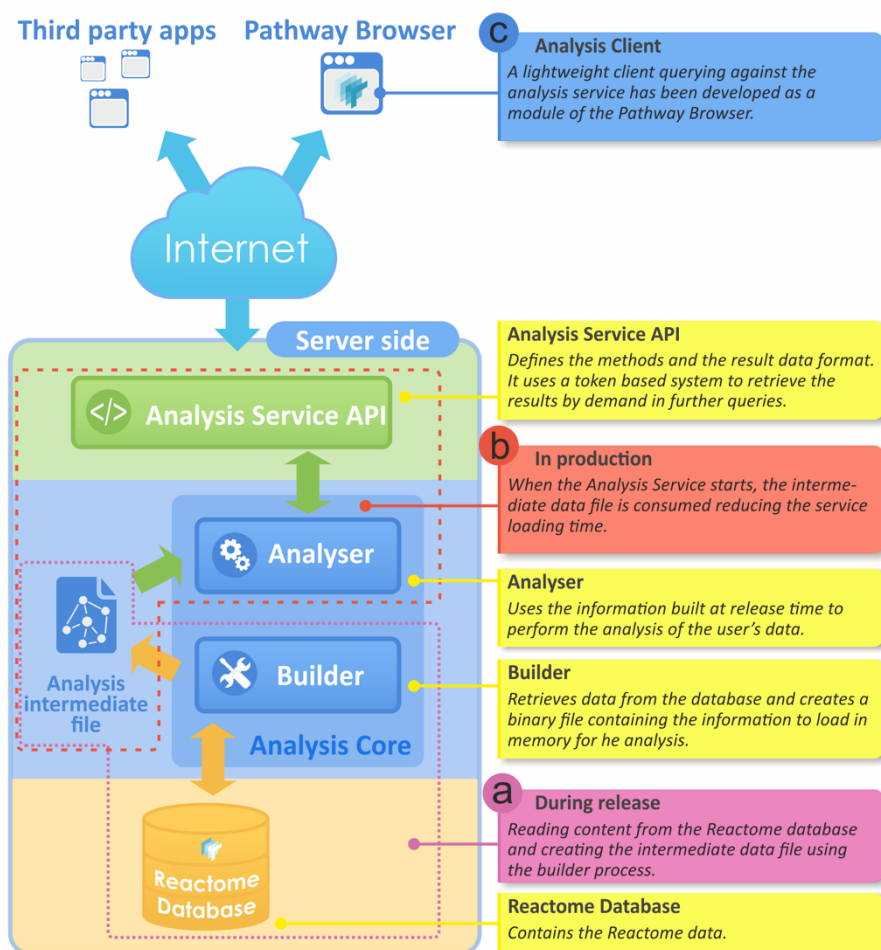
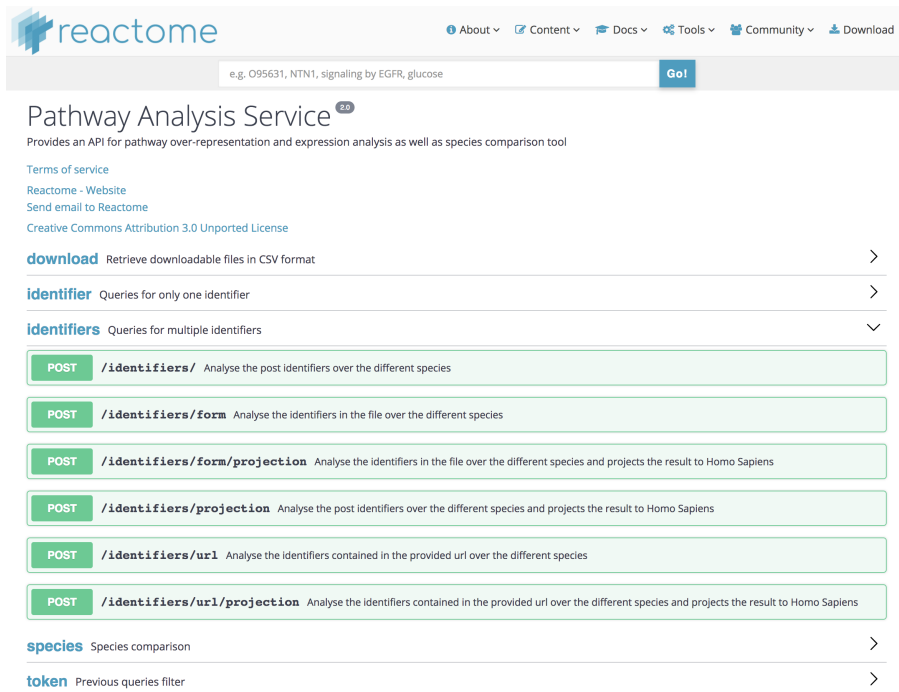


Figure 3.2. Analysis Service ecosystem. a) Reading content from the Reactome database and creating the intermediate data file using the builder process. b) When the analysis service starts, the intermediate data file is consumed reducing the service loading time. c) A lightweight client querying against the analysis service has been developed as a module of the Pathway Browser.

The Analysis Service was developed as a RESTful web service. REST stands for Representational State Transfer, which is an architectural style for networked hypermedia applications often used in web services that are

lightweight, maintainable, and scalable. A RESTful web service, also referred to as a RESTful API (Application Program Interface), defines a series of requests to GET, PUT, POST and DELETE data. Although REST is not dependent on any protocol, most RESTful services use HTTP as their underlying protocol.



The screenshot shows the Reactome Pathway Analysis Service RESTful API documentation page. The page features a search bar with the example query "e.g. O95631, NTN1, signaling by EGFR, glucose" and a "Go!" button. Below the search bar, the page title is "Pathway Analysis Service" with a version number "2.0". The main content area lists several API endpoints, each with a "POST" method and a brief description:

- POST /identifiers/** Analyse the post identifiers over the different species
- POST /identifiers/form** Analyse the identifiers in the file over the different species
- POST /identifiers/form/projection** Analyse the identifiers in the file over the different species and projects the result to Homo Sapiens
- POST /identifiers/projection** Analyse the post identifiers over the different species and projects the result to Homo Sapiens
- POST /identifiers/url** Analyse the identifiers contained in the provided url over the different species
- POST /identifiers/url/projection** Analyse the identifiers contained in the provided url over the different species and projects the result to Homo Sapiens

Additional sections include "species" for species comparison and "token" for previous queries filter. The page also includes navigation links for "download", "identifier", and "identifiers".

Figure 3.3. Reactome Analysis Service RESTful API documentation page (<https://reactome.org/AnalysisService/#/identifiers>).

All major development languages include frameworks for building RESTful web services. For the Analysis Service (<https://reactome.org/AnalysisService>), Java was chosen as the programming language with Spring MVC (<http://spring.io>) for the framework. API documentation (Figure 3.3) was integrated using OpenAPI, formerly known as Swagger v2.0 (<http://swagger.io>).

When an analysis is performed, a subset of the result is retrieved in order to improve the server-client data exchange time. To establish a mechanism whereby the client can introspect further through the details of a given analysis, a system based on tokens was enabled. Each analysis result has a different token that is assigned based on the user's input file MD5. This strategy helps in two aspects; (i) the same file won't be analysed twice during the same release cycle and (ii) collisions between different analysis tokens are prevented.

When a token is created, a serialised version of the result is kept on the server side, which requires hard drive space for file storage. To prevent excessive use of server side storage, the token's life is ensured for a minimum of 7 days but beyond this it enters into an LRU list, meaning that when the available space in the hard drive becomes limited, the result file associated with the least recently used token will be deleted and the token will be flagged as no longer available, leading to a prompting for the user to submit the sample again if the result is needed further. The token can be shared and allows later access through the API.

3.2.2. LIGHTWEIGHT CLIENT INTEGRATED INTO THE PATHWAY BROWSER

The Analysis Service was developed as a building block for Reactome pathways analysis, not only for third party users but also as the data analysis provider for a lightweight client integrated into the Pathway Browser.

The pathway analysis data submission interface, which is integrated into the Pathway Browser, can be launched by selecting the analysis button located in the top right corner of the Pathway Browser. User data is submitted by uploading a file or pasting content into the allocated text area (Figure 3.4).

The analysis is performed on the server side, with the results displayed in the Pathway Browser.

Analysis tools

Analyse your data

This tool merges pathway identifier mapping, overrepresentation and expression analysis into a single tabbed data analysis portal, with integrated visualization and summary features.

Select a file from your computer and click on the "GO" button to perform the analysis.

Select data file for analysis No file chosen Project to human

▼ Click here to paste your data or try example data sets...

Paste the data to analyse

#Probeset	10h_control	10h	14h	18h	24h
1053_at	8.044078	7.547358	6.706705	6.794622	7.475557
1729_at	6.869488	6.993184	7.129922	7.112222	7.04721
1861_at	6.437999	6.620892	6.20117	6.407735	5.717815
200000_s_at	9.381569	9.718002	9.874874	9.934639	9.499511
200001_s_at	12.555275	12.511845	12.566419	12.538042	12.439174
200003_s_at	12.401259	12.454083	12.275169	12.286342	12.415476
200005_at	9.609852	9.699299	9.73072	9.530097	9.194303
200012_x_at	12.486269	12.402275	12.302666	12.256543	12.282444
200014_s_at	10.371458	9.540578	9.978311	9.871472	8.753136
200016_x_at	12.110468	11.913288	11.938524	11.899243	11.458105
200022_at	12.205038	11.927471	12.064725	12.031422	11.932256
200023_s_at	10.377248	9.902753	9.990802	10.248432	9.513486
200024_at	12.651513	11.313608	12.422625	12.368137	12.170793
200032_s_at	12.364917	12.364901	12.563211	12.341325	12.278704
200034_s_at	11.802912	11.6947	11.817369	11.803903	11.527011
200035_s_at	12.392395	12.374603	12.46662	12.454694	12.308414
200039_s_at	10.485458	10.341833	10.429499	10.380328	9.810715

Some examples:

-
-
-
-
-
-
-

Project to human

Figure 3.4. The Analysis tools data submission interface, showing an example of the format used for multi-sample expression data (e.g. a time-series). Rows contain an identifier (probe set, gene name, etc.) in the first column. Subsequent columns contain numeric (expression) values for four time points, entered as tab-delimited text. The ‘project to human’ box at the bottom of the form, which is selected by default, causes any identifiers for non-human proteins in the data to be replaced by their human orthologs. Instructions for formatting data and lists of acceptable identifiers are provided in the user guide [Fabregat et al. 2015].

3.2.3. OVERLAYING ANALYSIS RESULTS IN THE PATHWAY BROWSER

A new tab of the Details Panel was developed to display analysis results in a tabular form (Figure 3.5). In addition, different components of the Pathway Browser are overlaid with analysis results. These are the Event Hierarchy tree, the Pathways Overview (Chapter 4) and the pathway Diagram Viewer (Chapter 5). In the case of the Event Hierarchy tree, there are two different

overlay cases: (i) for pathways, a suffix is added, detailing the number of hit entities and the number of entities contained in the pathway plus the FDR value for that pathway and (ii) for matched reactioning events, the name is boxed. Having the analysis results visually represented in the Pathways Overview widget, provides a birds-eye view of results that allows the user to navigate (zoom in) on areas of greatest interest. Selecting a row in the results table highlights the corresponding events in the hierarchy and focuses the Pathways Overview on the corresponding ‘burst’, or loads the corresponding pathway diagram. The analysis results overlay for the Pathways Overview and the Diagram Viewer is further explained in Chapters 4 and 5.

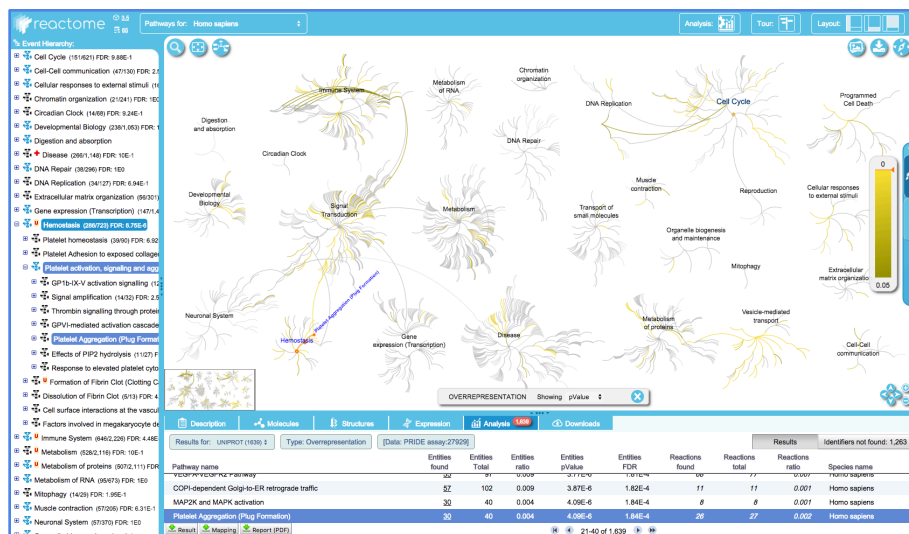


Figure 3.5. Analysis results for a PRIDE dataset to identify proteins over-expressed in activated human platelet releasate (assay 27929 <https://www.ebi.ac.uk/pride/ws/archive/protein/list/assay/27929.acc> in project PXD000072 <http://www.ebi.ac.uk/pride/archive/projects/PXD000072>). The results are overlaid on the different modules: the events hierarchy on the left, the Details Panel at the bottom and the viewport in the centre.

3.3. SUMMARY

Through the use of highly optimised, in-memory data structures and algorithms, a stable, high performance pathway analysis service has been developed to enable the analysis of genome-wide datasets within seconds, allowing interactive exploration and analysis of high throughput data. This was achieved by splitting the pathway analysis method in four steps, in a way that every challenge can be easily addressed in a polynomial time using the appropriate data structures, speeding up the process and minimising the memory usage so the whole data structure can be kept in memory for a high-performance analysis. The result is a new set of Analysis Tools which vastly improve Reactome analysis interface performance and stability.

This high-throughput pathway analysis is supported by a new RESTful web service interface (API), documented in detail (<https://reactome.org/AnalysisService/>), which allows use of the Reactome server for batch dataset analysis. Over-representation and expression data analysis can be performed against the Reactome database (`/identifier` and `/identifiers` methods) as well as species comparison (`/species` method). Once the data analysis or species comparison has been performed, a token is included in the client results allowing further service calls to retrieve more data related to the result (`/token` and `/download` methods). More information on how to use the analysis can be found in Reactome's developer zone at <https://reactome.org/dev/analysis/>.

A pathway analysis data submission interface was integrated into the Pathway Browser. The results of the analysis are displayed in a tabular form in a new tab integrated in the Details Panel. In addition, the results are overlaid in other modules of the Pathway Browser such as the Hierarchy Tree, the Diagram Viewer or the Pathways Overview.

The pathway analysis approach described in this chapter was deployed in the Reactome production web site, stably handling 78,827 analyses in 2015 and growing to handle 147,747 in 2016 and 733,988 in 2017 across 136,331 unique users. Memory usage for the Apache Tomcat running this service plus other services on the server side is set to 2 GB.

3.4. REFERENCES

Abiteboul,S. Hull,R.B. Vianu,V. (1995) Foundations of databases: The Logical Level 1st, Addison-Wesley Longman Publishing Co., Boston, Massachusetts, USA.

Benjamini,Y. and Hochberg,Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society, Series B, 57, 289-300.

Chowdhury,S. and Sarkar,R.R. (2015) Comparison of human cell signaling pathway databases--evolution, drawbacks and challenges. Database (Oxford).

Codd,E.F. (1972) In: Rustin,R. editor. Further normalization of the data base relational model, data base systems. Englewood Cliffs, Prentice-Hall.

De la Briandais,R. (1959) File searching using variable length keys. Proceedings of the Western Joint Computer Conference, San Francisco, California, 295-298.

Drăghici,S. Khatri,P. Martins,R.P. Ostermeier,G.C. Krawetz,S.A. (2003) Global functional profiling of gene expression. Genomics, 81, 98-104.

Fabregat,A. Sidiropoulos,K. Viteri,G. Forner,O. Marin-Garcia,P. Arnau,V. et al. (2017) Reactome pathway analysis: a high-performance in-memory approach. BMC Bioinformatics BMC series.18, 142.

-
- García-Campos,M.A. Espinal-Enríquez,J. Hernández-Lemus,E. (2015) Pathway analysis: state of the art. *Frontiers in Physiology*, 6, 383.
- Morrison,D. (1968) PATRICIA-Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM*, 15(4), 514-534.
- Reuter,J.A. Spacek,D.V. Snyder,M.P. (2015) High-throughput sequencing technologies. *Molecular Cell*, 58(4), 586-597.
- Shin,S.K. and Sanders,G.L. (2006) Denormalization strategies for data retrieval from data warehouses. *Decision Support Systems*, 42(1), 267-82.
- Talbi,E.G. and Zomaya,A.Y. (2008) *Grid Computing for Bioinformatics and Computational Biology*. Wiley, Chichester, UK.

4. PATHWAYS OVERVIEW

When the task of integrating the Analysis Service into the Pathway Browser was explored, it was considered important to provide a means to easily visualise the results and their significance as a genome-wide overview. In earlier implementations, results of the analysis were shown in a table-like representation or as a graphic representing the ontology-like hierarchical organisation of Reactome pathways [Petri et al. 2014].

Although both of these earlier methods represented analysis results, neither provided a complete picture that represented all areas of biology. For the table-like representation the order of the pathways was based on the statistics, instead of their parent-child relationship. In the case of the ontology-like tree visualisation, because the lower levels of the hierarchical organisation were hidden until expanded by the user, pathways located at higher positions in the tree with weak results could potentially mislead the user by not indicating the presence of highly significant ‘child’ pathways lower in the hierarchy.

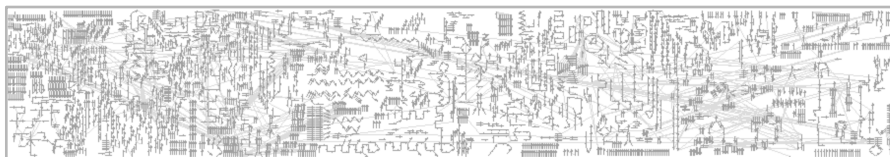


Figure 4.1. A reaction map view called “Starry Sky” because of its resemblance to the constellations at night. Decommissioned in 2011, it provided an interactive graphical representation of Reactome pathways. In this view a pathway is depicted as a set of interconnected arrows, each representing a reaction (e.g., "Formation of Cyclin E1:Cdk2 complexes"; "Phosphorylation of Cyclin E1:Cdk2 complexes"). Each of the reaction arrows is linked into pathways as dictated by the order of reactions in that pathway.

On early versions of the Reactome website, content was represented on the homepage as a graphical reaction map, commonly referred to as the ‘Starry Sky’ (Figure 4.1). This had a similar purpose to the Pathways Overview described later in this chapter, but used a completely different approach. The Starry Sky represented each reaction-like event, grouped by pathway. This view was also used to represent analysis results and was very popular with users, but it was laid out and maintained manually. As the number of pathways and reaction-like events increased, it became unmanageable and was eventually decommissioned [Croft et al. 2010]. Creating a graphical overview, capable of representing all pathways and their parent-child relationship in a way that could be perpetually extended with minimal manual intervention was one of the major requirements of the analysis redesign plan.

The popular notion of "a picture is worth a thousand words" is used to suggest that a complex set of ideas can be conveyed with just a single image, or that an image can convey meaning more effectively than a description. Following this guiding principle, finding an easy way to graphically represent Reactome pathways and their parent-child relationships is the first step to resolve the analysis result visualisation problem. This chapter focuses on the process of finding an appropriate layout and then explains the development of a lightweight web client to be included in the Pathway Browser to achieve the analysis result visualisation target.

4.1. FINDING AN APPROPRIATE LAYOUT

The Pathways Overview representation is a genome-wide, hierarchical visualisation of pathways showing their parent-child relationships in a space-filling graph. As Reactome is released quarterly, content is unaltered between

releases, so the layout for each species represented in Reactome can be generated once per release and then reused as needed.

This section discusses the initially proposed solution, which was based on a force-directed layout approach. Consequent problems, caveats, requirements and specification are described, finishing with a description of a custom deterministic layout algorithm created to address the problem in a future-proof, scalable and maintainable manner.

4.1.1. FORCE-DIRECTED LAYOUT APPROACH

The first step towards designing the Pathways Overview was exploring the options to reuse an existing forced-directed layout algorithm, to assess whether the results fit the requirements. Force-directed layout algorithms are a class of graph drawing algorithms that produce an aesthetically pleasing result. These algorithms do not rely on contextual information; they are based solely on information contained within the graph structure [Kobourov 2012]. A simple use-case of force-directed algorithm would use repulsive forces between detached nodes and attractive forces between connected nodes.

Force-directed layout algorithms are implemented in many Java libraries such as yFiles (<https://www.yworks.com/products/yfiles-for-java-2.x>), GRAD (<https://www.gradlibrary.net/>) or Gephi Toolkit (<https://gephi.org/toolkit/>). The first of these, yFiles, is a commercial library, consequently it was discarded in favour of an open source option. Gephi Toolkit offers a well-documented API (<https://github.com/gephi/gephi/wiki/How-to-code-with-the-Toolkit>) that allows graph creation using custom data and facilitates the execution of layout algorithms for a given set of conditions.

Amongst the available layout algorithms, Gephi implements Force Atlas, a specific type of force-directed layout used for real-world networks [Bastian et

al. 2009] and ForceAtlas2, an improved version of the latter that can handle large networks while retaining high quality. With ForceAtlas2 node repulsion is approximated with a Barnes-Hut calculation, which reduces algorithm complexity by replacing the attraction and repulsion forces with a scaling parameter [Jacomy et al. 2014].

To use the Force Atlas layout algorithm with Reactome data, all human pathways and their parent-child relationships were extracted and a graph instance created using the Gephi API. Options such as the duration of the algorithm can be easily set up with the AutoLayout object. Figure 4.2 shows the result of applying the Force Atlas layout to the Reactome human pathways graph for different durations using the option to locate all nodes in the same position at the start.

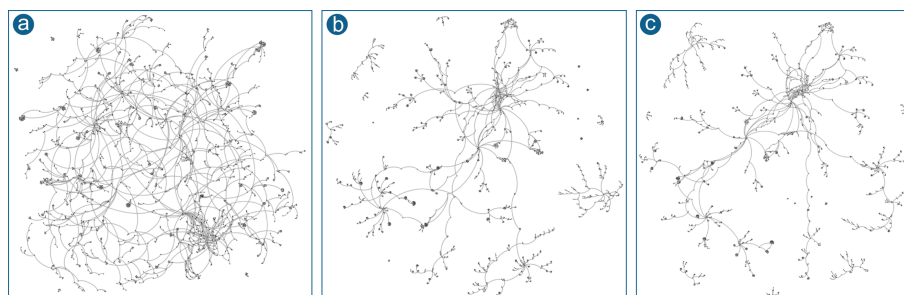


Figure 4.2. Force Atlas layout algorithm applied to the human pathways in Reactome (release 51). Duration of the execution of the algorithm from left to right is 1, 5 and 10 minutes respectively. Nodes corresponds to a human pathway and each edge indicates a parent-child relationship. For the three cases, all nodes starting position is the same to start applying the Force Atlas layout.

Applying the Force Atlas layout algorithm, the first observation is that while the algorithm stabilises after a short amount of time, a slightly longer execution time results in a clearer separation between the different biological

areas, or top-level pathways as named in Reactome (Figure 4.2 b and c). Consequently, the different related areas of biology, that were already connected with one or more edges, remain closer due to the topology of the graph being laid out. A problem with this method is that, due to the nature of the algorithm, each execution even when run for the same duration, produces a different result.

The disadvantage of having slightly different results at each release becomes clearer when considering use cases such as the comparison of data analysis results between releases. Using a geographical map analogy, it is expected that a country will always be in the same location on the map. Extending this analogy to pathway data, it would be more intuitive for the user if pathways remain in the same location from one release to the next.

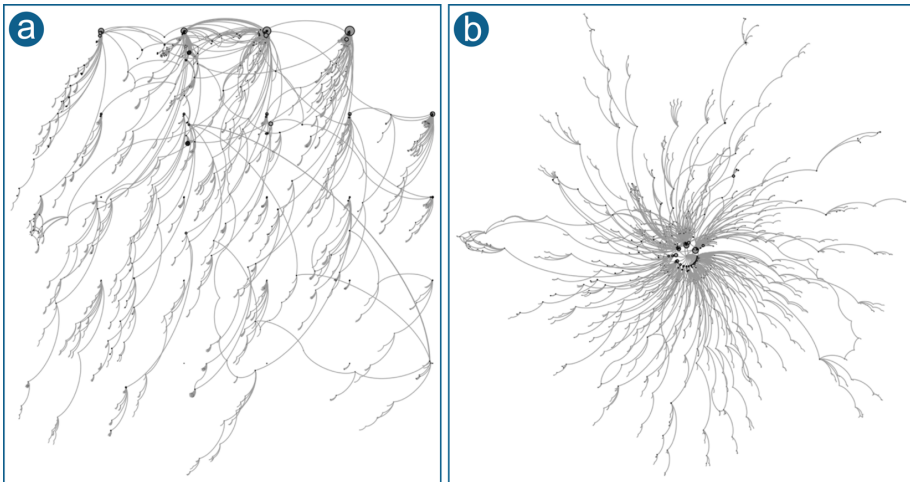


Figure 4.3. Force Atlas layout for a duration of 10 minutes. (a) The top-level pathways are fixed to a grid layout. (b) The top-level pathways are fixed to circumference in the inner ring. Each node in every image corresponds to a pathway and the edges indicate the parent-child relationship.

A first approach to overcome this variability in location of the main areas of biology was to try the option available for the Force Atlas layout algorithm that prevents a set of nodes being moved during execution. Configuration of this starting scenario consisted of fixing the location of all top-level-pathways organised in a grid layout, making them static so the layout algorithm could not move them from their original location. The rationale behind this approach is to keep together in the same area the connected nodes that represent subpathways. Figure 4.3a shows the result for this case after execution of the algorithm for 10 minutes.

The organisation of top-level-pathways in a grid layout (Figure 4.3a) introduces a constraint that results in crossing edges and therefore works against the principle of force-directed layouts, which are designed to reduce this. As an alternative, the top-level-pathways were arranged in a radial organisation. The results of this second scenario are presented in Figure 4.3b, illustrating the problems associated with this alternative, namely that the centre of the view is crowded and difficult to examine while space in the corners is underutilised.

After this exploratory phase using an existing force-directed layout algorithm, a better understanding of the problem was acquired and an improved set of requirements were established. The requirements are exposed in subsection 4.1.2 and a custom layout algorithm to fulfil them is presented in 4.1.3.

4.1.2. LAYOUT REQUIREMENT

Based on the results of the previous work, a new set of requirements for an appropriate layout were determined, covering two main targets: (i) the layout must graphically display the parent-child relationships present in Reactome's pathway hierarchy and (ii) provide an easy way to overlay analysis results that

facilitates comprehension. For the first of these requirements, a more detailed specification is as follows:

- The view has to be structured, easy to follow and intuitive for the final user.
- The size of each node should be proportional to the size of the represented pathway.
- There must be only one node for each pathway.
- Since the pathways parent-child relationships forms a graph, the crossing edges should be minimised.
- Nodes should keep their relative positions after each release. When new sibling nodes are introduced (new branches added to the hierarchical tree), existing nodes should remain as close as possible to their previous positions.

4.1.3. CUSTOM RADIAL LAYOUT DEFINITION

The result in section 4.1.1 was not satisfactory because the layout algorithm did not make good use of the available space and there were too many crossing edges. To overcome these problems, a second implementation of the layout algorithm was developed where each top-level-pathway was manually laid out, separately arranged to optimise the use of the viewport space and minimise the number of crossing edges.

Based on the experience with a force-layout algorithm, adopting a radial layout seemed an appropriate way forward. Using this type of layout, each concentric ring could be used to place nodes representing subpathways of the direct inner ring. Conventional radial tree layouts [Eades 1992; Melançon & Herman 1998; Teoh & Herman 1998; Wills 1999] did not satisfactorily meet the full set of requirements, suggesting that new constraints are required. A

custom radial layout was developed and applied in two different ways. In the first, a generic case was created to lay out all top-level-pathways in one ‘burst’.

In this section, a pathway node is defined as P_i^l where l is its level in the events hierarchy (or the ring in the layout, e.g. 1 for top-level-pathway or 2 for children of a given top-level-pathway) and i is the position among its siblings. Common to both phases is the definition of the size of a node when laid out in the viewport. Following the requirements, the size of each node has to be proportional to the size of the represented pathway, so for a given pathway i , its ratio R_i is defined as:

$$R_i = \frac{S_i}{S}$$

Where $i \in [1, p]$ being p the number of pathways in the pathway species, S is the number of unique entities annotated in that species, known as species background, and S_i the number of unique entities annotated in the pathway represented by the node P_i . Using the ratio of a given pathway i , its radius r_i is:

$$r_i = R_i F$$

Where F is a factor to be chosen in the final view to artificially alter all the nodes size for aesthetic reasons. Note that the size of a node representing a pathway is calculated in the same way for all nodes regardless of their position in the viewport. The method used to calculate ratios varies, based for the location of the node within the layout, as explained in detail below.

Scenario 1: All top-level-pathways in a single burst

The areas of biology represented in Reactome, as well as their associated hierarchical subpathway branches, differ significantly in size, in terms of the number of unique entities they contain and the number of hierarchical levels. When a common radial layout algorithm is used, this tends to result in some very crowded areas while other areas are relatively empty.

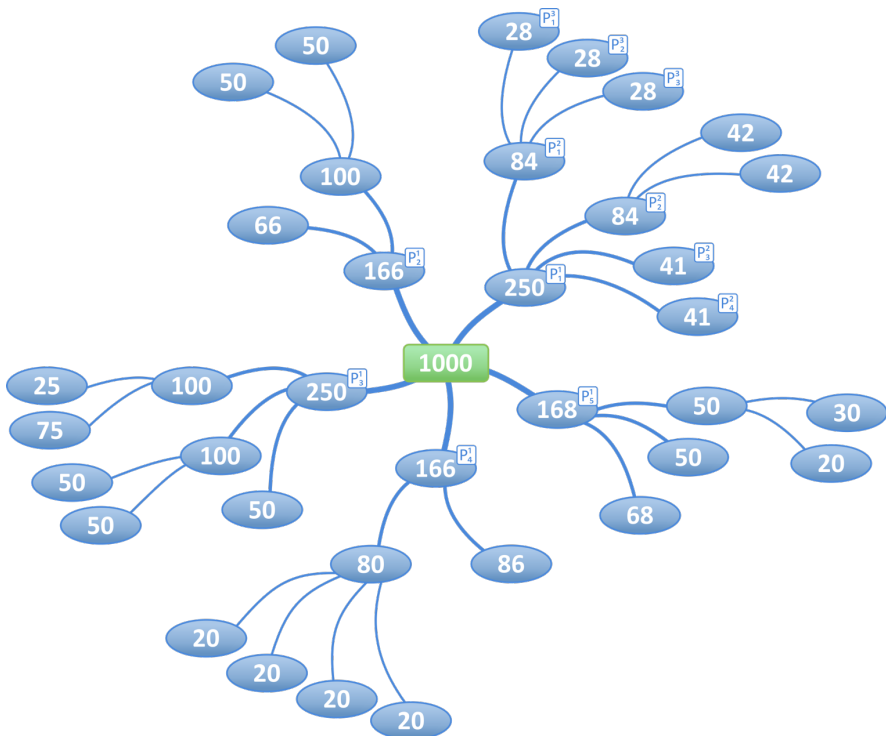


Figure 4.4. Hypothetical example of pathways and their relationships for scenario 1. The centre node represents the unique entities curated in the species. Each node represents a pathway and the number in it represents its number of unique curated entities. Edges represent their parent-child relationship where the outermost is child of the innermost. Nodes labelled with P_i^j are the building blocks for figures 4.5, 4.6, 4.7 and 4.9.

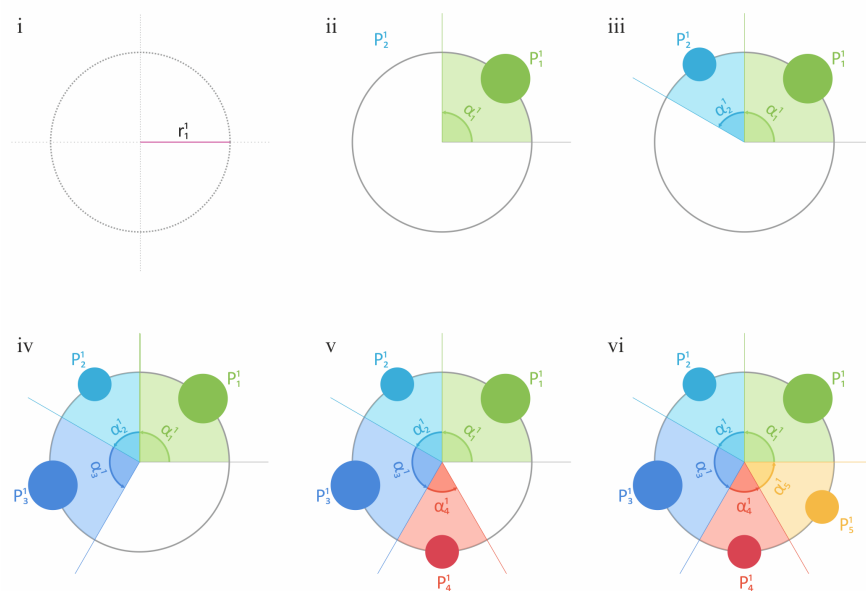


Figure 4.5. Schematic representation of the construction of the customised radial layout for the first level of Figure 4.4 containing 5 hypothetical top-level-pathways. In (i) the radius is defined. From (ii) to (vi) a top-level-pathway area is reserved at the time by calculating the angle based on its ratio.

To address this problem, the available space should be partitioned according to the relative size of the pathway, which as seen above is its ratio R_i . In this scenario the aim was to represent all pathways in the same view, laid out as a single radial layout where the top-level pathways in Reactome form a circle, with branches radiating from them to represent lower hierarchical levels.

To reinforce the explanation of the custom layout algorithm, figures 4.5, 4.6, 4.7 and 4.9 refer to the hypothetical pathways example shown in Figure 4.4 instead of a real case using Reactome data. In scenario 1, top-level-pathways are represented in the first ring (Figure 4.5), subsequent hierarchical levels are placed in concentric rings radiating outwards from the centre of the circle,

based on their parent-child relationships (Figure 4.6). Consequently, the centre of the layout does not contain a node rather it conceptually represents all of biology for the given species.

The angle associated with each top-level-pathway P_i^1 is based on its ratio R_i^1 which is calculated as its number of unique curated entities S_i^1 divided by the number of unique curated entities in its species S :

$$R_i^1 = \frac{S_i^1}{S}$$

Due to the nature of its construction, the sum of the ratios for all top-level-pathways equals one:

$$\sum_{i=1}^t R_i^1 = 1$$

Where t is the number of top-level-pathways. Once the ratio for each top-level-pathway R_i^1 is defined, their angle α_i^1 is calculated as the proportion of 2π by the ratio R_i^1 :

$$\alpha_i^1 = \frac{2\pi}{R_i^1}$$

Similarly to the ratios, the sum of the angles for all top-level-pathways cover the circumference:

$$\sum_{i=1}^t \alpha_i^1 = 2\pi$$

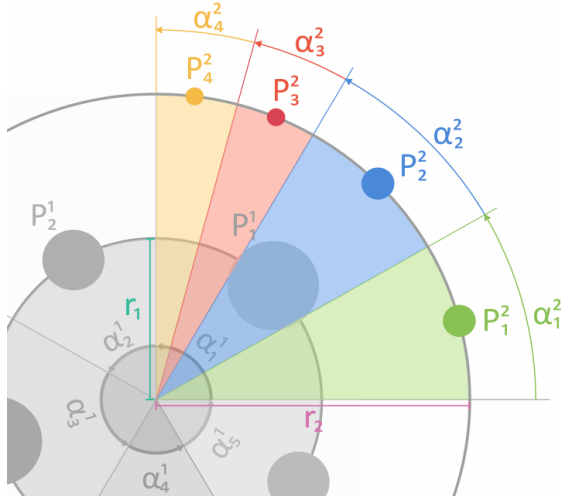


Figure 4.6. Schematic representation of the construction of the second ring of the customised radial layout where the area for each of the children of a top-level-pathways is reserved based on its ratio relative to their parent.

Assuming that A_j^{l-1} is the ancestor of a given pathway P_j^l (where $l > 1$) and ψ_j^{l-1} is the number of unique curated entities it contains and ϕ_j^{l-1} is its assigned angle, then the method to calculate the ratio θ_j^l of P_j^l varies from the original R because it must split the available ancestor angle ϕ_j^{l-1} :

$$\theta_j^n = \frac{S_j^n}{\psi_j^{n-1}}$$

In this case, as it happens for the top-level-pathways case, the sum of the ratios for all children equals one:

$$\sum_{j=1}^m \theta_j^n = 1$$

Where m is the number of Λ_j^{l-1} children. Once the ratio θ_i^1 is known, the angle α_i^1 for the node representing each child of pathway P_i^{n-1} , is calculated as the proportion of ancestor angle ϕ_j^{n-1} divided by its ratio:

$$\alpha_j^n = \frac{\phi_j^{l-1}}{\theta_j^n}$$

In this case, the sum of the angles for all Λ_j^{l-1} children equals the parent assigned angle ϕ_j^{l-1} :

$$\sum_{j=1}^m \alpha_j^n = \phi_j^{l-1}$$

A further calculation defines the location of each node, but before presenting this, it is important to explain the radius requirements for each ring. To make the final view more appealing, each ring needs sufficient space to clearly differentiate the level. To accomplish this, a fixed radius can ensure a clear differentiation but in some cases cannot be used for deeper levels because the available arc angle for each sets of siblings is reduced at each level. Consequently, there can be scenarios in which there is insufficient space to render the nodes in the arc, using a fixed node radius. In these cases the radius must be increased to allocate enough space to render all the required nodes without overlap. The radius for each level r_l is calculated as:

$$r_l = \frac{\sum_{i=1}^m 2 r_i^l}{\phi_j^{l-1}} \quad \text{if } r_l < r \text{ then the used radius is the default one } r$$

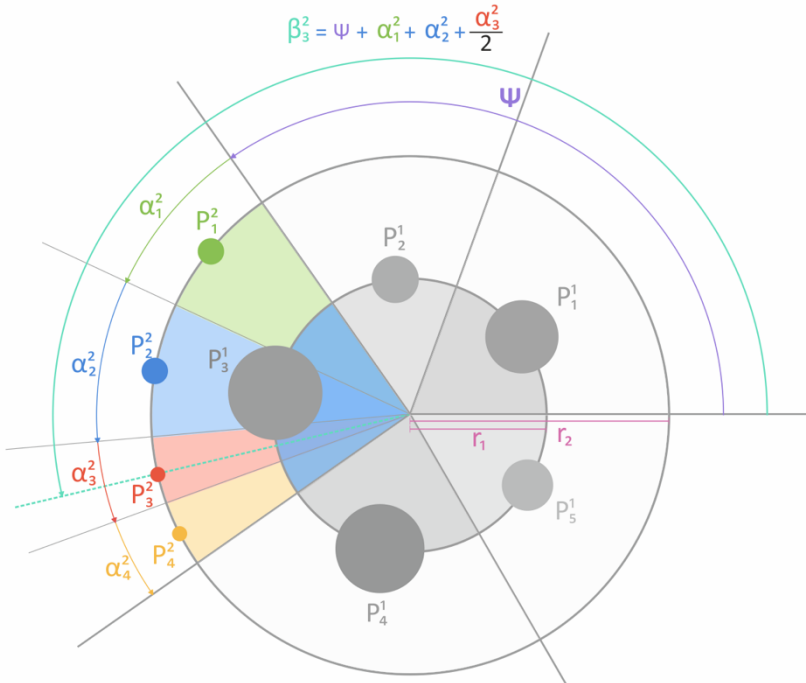


Figure 4.7. Represents the logic behind the calculation of centre of the node to be added to a given ring.

The final step is to calculate the point at which a given pathway node P_i^n is laid out (Figure 4.7). This is calculated differently depending on whether the node is present in the first ring or in another. If in the first ring, the angle β_i^n is calculated depending on the value of i :

$$\beta_i^n = \frac{\alpha_i^n}{2} \quad \text{when } i = 1$$

$$\beta_i^n = \sum_{j=1}^{i-1} \alpha_j^n + \frac{\alpha_i^n}{2} \quad \text{when } i > 1$$

Where the node is not in the first ring, the accumulative arc inherited from the parent previous siblings Ψ must be taken into account. The formula used

depends on the value of i , such that for i greater than 1, the angle of the previous sibling must also be taken into account:

$$\beta_i^n = \Psi + \frac{\alpha_i^n}{2} \quad \text{when } i = 1$$

$$\beta_i^n = \Psi + \sum_{j=1}^{i-1} \alpha_j^n + \frac{\alpha_i^n}{2} \quad \text{when } i > 1$$

Then its node location point (L_x, L_y) is calculated based on the center coordinates (C_x, C_y) and the angle β_i^n as:

$$L_x = C_x + r_n \cos(\beta_i^n)$$

$$L_y = C_y + r_n \sin(\beta_i^n)$$

Where r_n is the radius of level for current pathway node.

Once the nodes are laid out, the next step is to connect them following their parent-child relationship. When the lines representing this relationship are drawn, the two main options to consider are (i) linear curves or (ii) quadratic curves. Linear functions are typically in the form of $y = mx + b$ where m stands for the slope, or rate of change, and b is the y intercept. These are graphed as straight lines because the x variable is not raised to any exponent. Quadratic functions are typically in the form $y = ax^2 + bx + c$ that always have the x variable to the second power, or in other words, the x is squared. This makes for a symmetrical, curved graph called a parabola [Hughes-Hallett et al. 2013].

Even though linear curves are easier to compute and therefore to render, there are cases where the node size cannot be differentiated from the edge width when these are used, so the final users might not perceive it appropriately

(Figure 4.8a). This situation can be easily avoided by the usage of quadratic curves that form shapes similar to parabolas (Figure 4.8b).

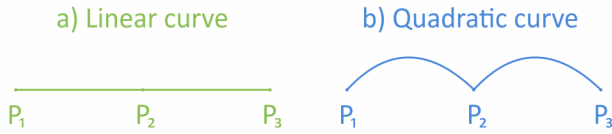


Figure 4.8. Quadratic functions to render lines joining parent-child related nodes vs linear functions.

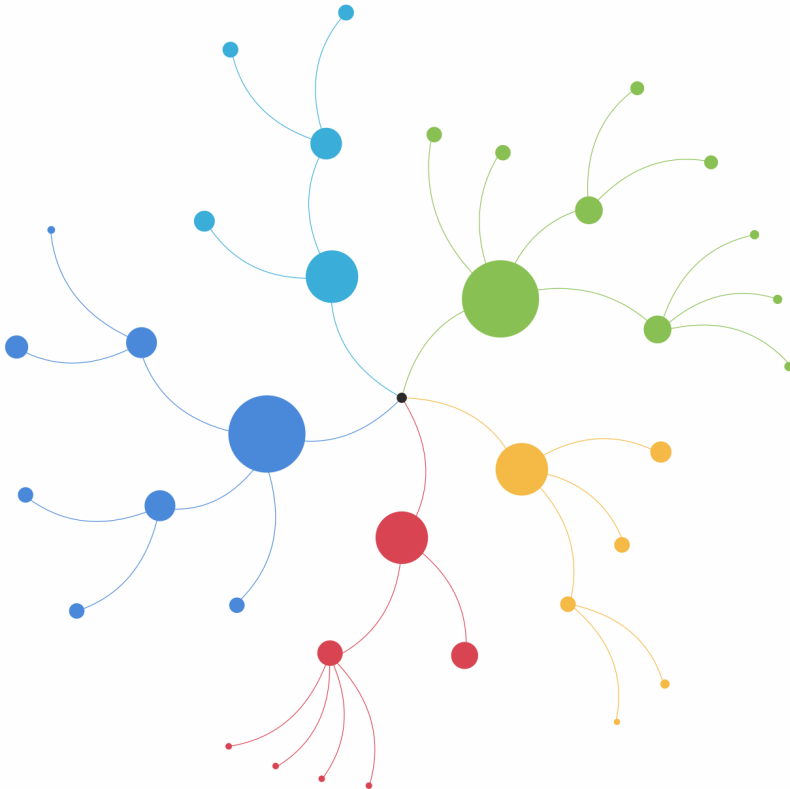


Figure 4.9. Result of applying the custom radial layout algorithm to the hypothetical example of pathways and their relationships for scenario 1 presented in Figure 4.4.

Quadratic curves, more specifically Bezier curves, were chosen to represent pathway parent-child relationships due to their smooth shape in the final result of the custom radial layout (Figures 4.9 and 4.10). Bezier curves are parametric smooth curves generated from the two end points, corresponding with the centre of the connected nodes and one or more control points, which may not necessarily fall on the curve, used to calculate the path of the curve via interpolation [Bartels et al. 1998]. Figure 4.9 exemplifies the result of applying the custom radial layout algorithm for the hypothetical case shown in Figure 4.4; Figure 4.10 shows all the human pathways in release 51.

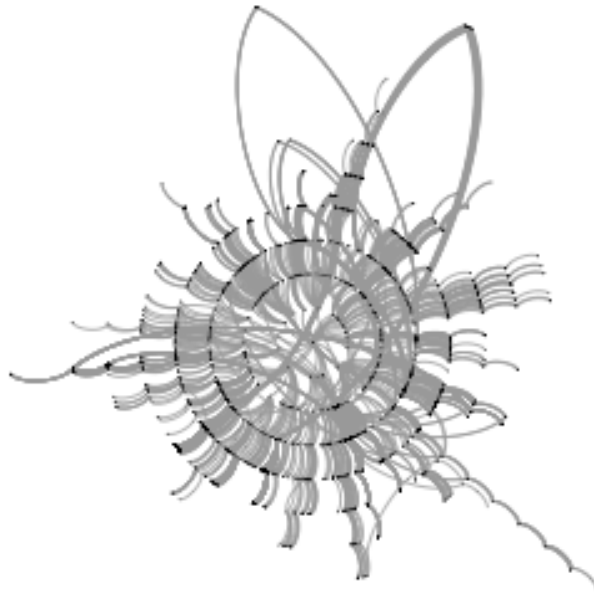


Figure 4.10. Pathways Overview using the custom radial layout for scenario 1 including all human pathways for release 51. All the top-level-pathways populate the first ring (release 51).

Scenario 1 results in a very crowded layout in the centre of the image that, at the same time, has a lot of empty space in the corners. This causes a lot of

edge crossing, which goes against the requirements set in section 4.1.2. In order to optimise the usage of the available space, a second scenario is contemplated in which a burst will be created per top-level-pathway.

Scenario 2: One burst per top-level-pathway

To overcome the problems found with the result of scenario 1, the idea for scenario 2 was to apply the same algorithm to a number of bursts where a top-level-pathway node was setup in the centre. In practise, the procedure is the same as for the previous scenario, but taking the number of unique entities curated in each top-level-pathway as the background for the first ring of each burst, instead of the number of unique entities curated in the species.

Since each top-level-pathway node's initial location is set up manually, the layout algorithm can now be classified as semi-automatic. Figure 4.11 presents the result for scenario 2 where the initial position of each top-level-pathway node was set in a regular grid. This approach optimises the space usage although the initial distribution causes a lot of edge crossing. Most of the edge crossing could be avoided with a cleverer organisation of the initial top-level-pathways arrangement.

Apart from the capability of setting up the location of each top-level-pathway node, other useful options to avoid edge crossing that were added to the layout algorithm are (i) the starting angle and (ii) the direction for which the algorithm lays out the children (e.g. clockwise or anticlockwise). After carefully applying all the settings for each node (top-level-pathway node location, starting angle and direction) the result for all human pathways in release 52 is shown in Figure 4.12.



Figure 4.11. First attempt using the custom layout algorithm per top-level pathway (release 51). Only edges are rendered. Nodes arranged in a grid following alphabetical order causing too much edge crossing (Scenario 2).

The custom layout algorithm applied in the scenario 2 covers all the requirements enumerated in subsection 4.1.2 because (i) it produces a structured view that seems easy to follow and intuitive for the final user, (ii) the size of each node is proportional to the size of the represented pathway, (iii) there is only one node for each pathway, (iv) the number of crossing edges is minimised, although its optimisation cannot be ensured due to the human intervention component of the semi-automatic approach and (v) nodes are likely to keep their relative positions after each release based on the deterministic approach in which the layout algorithm has been developed.

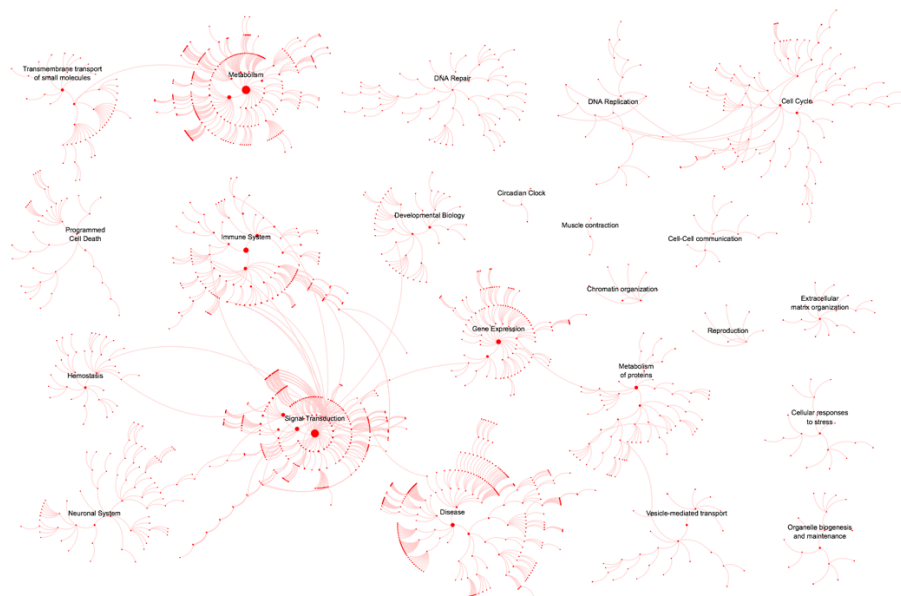


Figure 4.12. Final representation of the Pathways Overview using the custom radial layout for each top-level-pathway for release 52. The location, starting angle and orientation for each top-level-pathway node was set manually aiming to minimise the edge crossing. The name added next to each burst centre indicates the represented area of biology.

Laying out the inferred pathways for other species

Previous subsections focused on the human pathway nodes layout. However, during each quarterly release, Reactome content is computationally inferred to different, evolutionarily divergent eukaryotic species for which high-quality whole-genome sequence data are available (<https://reactome.org/documentation/inferred-events>).

The computationally inferred pathways in other species represent a subset of those in human, so when the Pathways Overview is generated for other inferred species, the pathways nodes should remain in the same location as

the layout algorithm positioned their human orthologs. To achieve the described result, each human pathway node position is used for the node representing the inferred pathway node in the target species.

4.1.4. STORING THE RESULT OF THE CUSTOM RADIAL LAYOUT

With each new data release, the custom radial layout algorithm is executed and a different file is generated per species. Thus, it must be stored to be used every time the Pathways Overview is rendered.

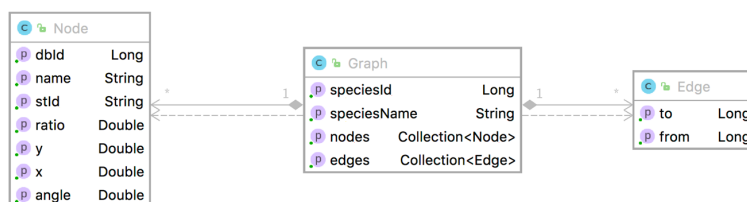


Figure 4.13. UML class diagram for the Pathways Overview layout data container. The Node class contains information related to each pathway node such as the identifier, name, ratio or its location (x,y). The Edge class contains the parent-child relationships between nodes. The Graph class contains information related to the species of the represented Pathways Overview and aggregates the Node and Edge instances.

The required information to render the Pathways Overview are the collection of nodes and their parent-child relationship (Figure 4.13). The essential data for each node, apart from the represented pathway identifier and name, are the location (x,y) for the render algorithm to know where to locate it and a parameter to define its size (in this case the ratio).

When the model in Figure 4.13 is populated, the next step is serialising it to a file in a specific format. As discussed in the publication for the Reactome Diagram Viewer [Fabregat et al. 2017], the JavaScript Object Notation (JSON - <http://www.json.org>) is less verbose than the eXtensible Markup Language

(XML) format (<https://www.w3.org/TR/REC-xml>) and thus has a smaller footprint that helps reducing client loading time. Moreover, JSON's natural mapping to JavaScript objects is faster and uses fewer resources than its XML counterpart [Lin et al. 2012; Nurseitov et al. 2009; Wang 2011]. Therefore JSON was chosen; an example of the Pathways Overview layout content for human is shown in Figure 4.14.

```

{
  "speciesId": 48887,
  "speciesName": "Homo sapiens",
  "nodes": [
    {
      "dbid": 196741,
      "stid": "R-HSA-196741",
      "name": "Cobalamin (Cbl, vitamin B12) transport and metabolism",
      "ratio": 0,
      "x": 533.12,
      "y": 313.32,
      "angle": 3.52
    },
    {
      "dbid": 983170,
      "stid": "R-HSA-983170",
      "name": "Antigen Presentation: Folding, assembly and peptide loading of class I MHC",
      "ratio": 0.01,
      "x": 294.05,
      "y": 87.76,
      "angle": 3.31
    },
    {
      "dbid": 983169,
      "stid": "R-HSA-983169",
      "name": "Class I MHC mediated antigen processing & presentation",
      "ratio": 0.03,
      "x": 319.92,
      "y": 83.52,
      "angle": 3.49
    },
    ...
    {
      "dbid": 1834949,
      "stid": "R-HSA-1834949",
      "name": "Cytosolic sensors of pathogen-associated DNA ",
      "ratio": 0.01,
      "x": 370.84,
      "y": 147.64,
      "angle": 1.45
    }
  ],
  "edges": [
    {
      "from": 3296469,
      "to": 5683329
    },
    {
      "from": 1839126,
      "to": 8851708
    },
    {
      "from": 351202,
      "to": 1237112
    },
    {
      "from": 2644603,
      "to": 2691230
    },
    {
      "from": 1430728,
      "to": 15869
    },
    ...
  ]
}

```

Figure 4.14. JSON example for release 63. It does not show all the file content but a representative part of it where nodes and edges are represented. The file is generated once per release and is available in the download section at https://reactome.org/download/current/fireworks/Homo_sapiens.json.

4.2. LIGHTWEIGHT WEB BROWSER WIDGET

Finding an appropriate layout for the Pathways Overview was the first of a two-step process. The second step was providing a web browser widget to display the overview and allow users to interactively navigate through the pathways in a graphical manner. The widget was originally conceived to be integrated as a module in the Pathway Browser.

One goal of the widget was to offer a mechanism for overlaying pathway analysis results, as a means to offer users a first-glance tool that helps identify the areas of biology that were significantly represented in a given query. Details of the widget are explained in section 4.2.1, while the capability of overlaying pathway analysis results is discussed in section 4.2.2.

4.2.1. WIDGET DETAILS

The widget was developed for inclusion as one module of the Pathway Browser. Thus, it was written using Java and the GWT toolkit (Section 2.2). The source code and documentation are available in GitHub at <https://github.com/reactome-pwp/fireworks>. To be reused by third parties, a JavaScript version was needed to be easily included in their sites. To address this, a JavaScript wrapper was developed so the transpilation produces a result that can be easily reused in JavaScript. The wrapper is available at <https://github.com/reactome-pwp/fireworks-js> and its API documentation is at <https://reactome.org/dev/pathways-overview/js>.

When the widget was first developed, as presented in this thesis, it did not include all the features that it has now. For example, the integrated Solr-based search against all Reactome content and the right-hand side context menu were added later. These features were developed together with other members of the Reactome developer team and therefore are not included in this section.

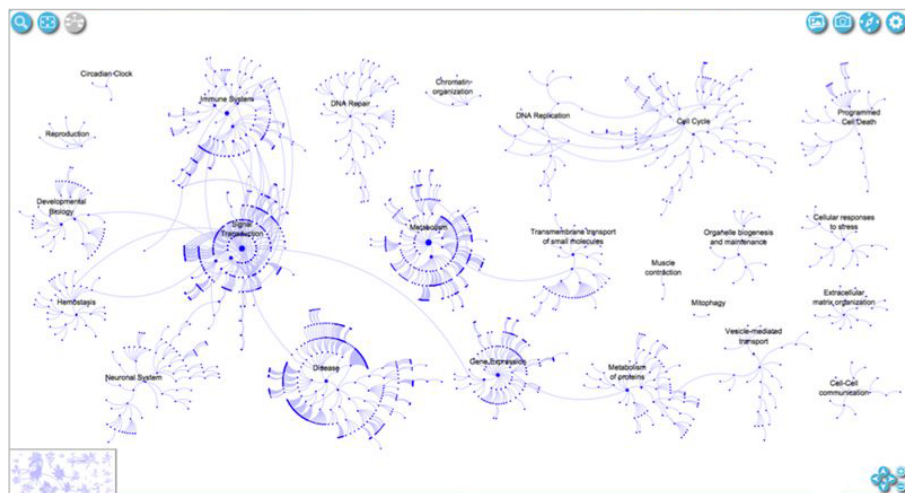


Figure 4.15. Pathways Overview widget representing the pathways contained in the data release version 54 (Figure 1 B in Fabregat et al, 2015).

Figure 4.15 shows the Pathways Overview widget. Controls in the bottom right corner allow movement of the viewport content up/down/left/right as well as zooming in and out. The zoom effect can also be achieved using the mouse scroll wheel or similar touchpad actions. When zooming is performed with the mouse wheel, the action occurs relative to the position of the mouse and captures the scrolling speed such that a strong movement of the wheel rapidly accelerates zooming to quickly show the final zoom level. Slower mouse wheel scrolling produces small changes of the level of zoom.

Depending on the level of zoom, more or less information is shown on the screen. As the zoom level increases, pathway names appear alongside the pathway nodes, as sufficient space becomes available to avoid an overcrowded view at lower levels of zoom (Figure 4.16). Names are rendered starting from the outside edge of the ring of nodes oriented to follow the angle of association of the node to its parent. This strategy avoids overlapping text,

optimises the use of space and clearly distinguishes to which node the name is assigned.

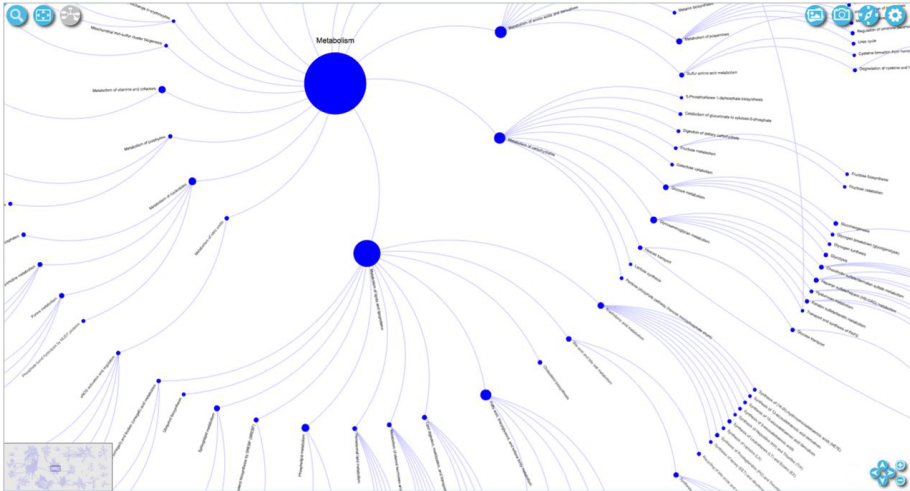


Figure 4.16. A zoomed-in view of the Metabolism burst showing individual subpathway groups (Figure 1 C in Fabregat et al. 2015).

Controls in the top left corner from left to right allow the user to perform the following functions: search; set the zoom level to fit the diagram to the available area; open the diagram for the pathway represented by the selected node. The latter action of opening the diagram for the pathway represented could also be achieved by double-clicking on the node or its pointing edge. The search, as it was originally developed, only supported the use of pathway names and identifiers (Figure 4.17).

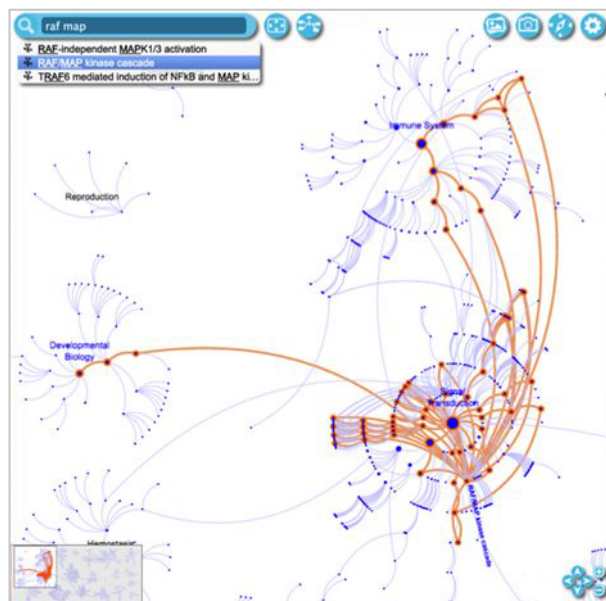


Figure 4.17. The RAF/MAP kinase cascade pathway is highlighted to show its involvement in multiple bursts (Figure 1 A in Fabregat et al. 2015).

In this graphical representation, when a specific pathway like RAF/MAP kinase cascade is present in more than one branch of Reactome’s hierarchy of pathways, it is displayed as shared by more than one burst by the inclusion of an arc that connects its representing nodes between bursts. When any of its nodes are selected, all of them, as well as their parents up to the burst centres are highlighted (Figure 4.17).

The controls on the top right corner starting from the left-hand side allow the user to (i) display the associated illustration when available for the selected node, (ii) save a snapshot of the current view, (iii) display the Pathways Overview key and (iv) display an options menu where colour profiles for the display and overlay can be selected.

When the selection of a pathway is performed outside the widget and the corresponding method in the API is called to update the status, the widget smoothly updates the view. The update consists of moving the selected node into the viewport and setting the appropriate level of zoom to display all the parent nodes. The time and speed of movement depend on the euclidean distance from the centre of the current view to the position and area of the target node. The transition movement is linear and coincides with the zoom.

The thumbnail in the bottom left corner interactively updates its content as actions take effect in the main display. When the view is zoomed-in, the thumbnail shows the visible region as a box. Moving this box in the thumbnail changes the region displayed in the main viewport (Figure 4.17).

To develop this widget, some techniques used in the gaming industry were studied and applied. The multi-layer HTML5 canvas technique was included to improve visual feedback while a QuadTree data structure was used to rapidly identify when the mouse pointer is hovered over a node or edge [Agarwal and Erickson, 1998]. These techniques were also used in the Diagram Viewer implementation (Chapter 5 - Section 5.2.1) and are discussed in detail in one of the papers that support this thesis: “Reactome diagram viewer: data structures and strategies to boost performance” [Fabregat et al. 2017].

4.2.2. *OVERLAYING ANALYSIS RESULTS*

The Pathways Overview widget has a module that integrates the Reactome Analysis Service. Analysing data produces an analysis token (Section 3.2.1) that can be set so the result is displayed as an overlay on top of the Pathways Overview. Nodes and edges are coloured according to their relevance in the analysis results [Fabregat et al. 2015].

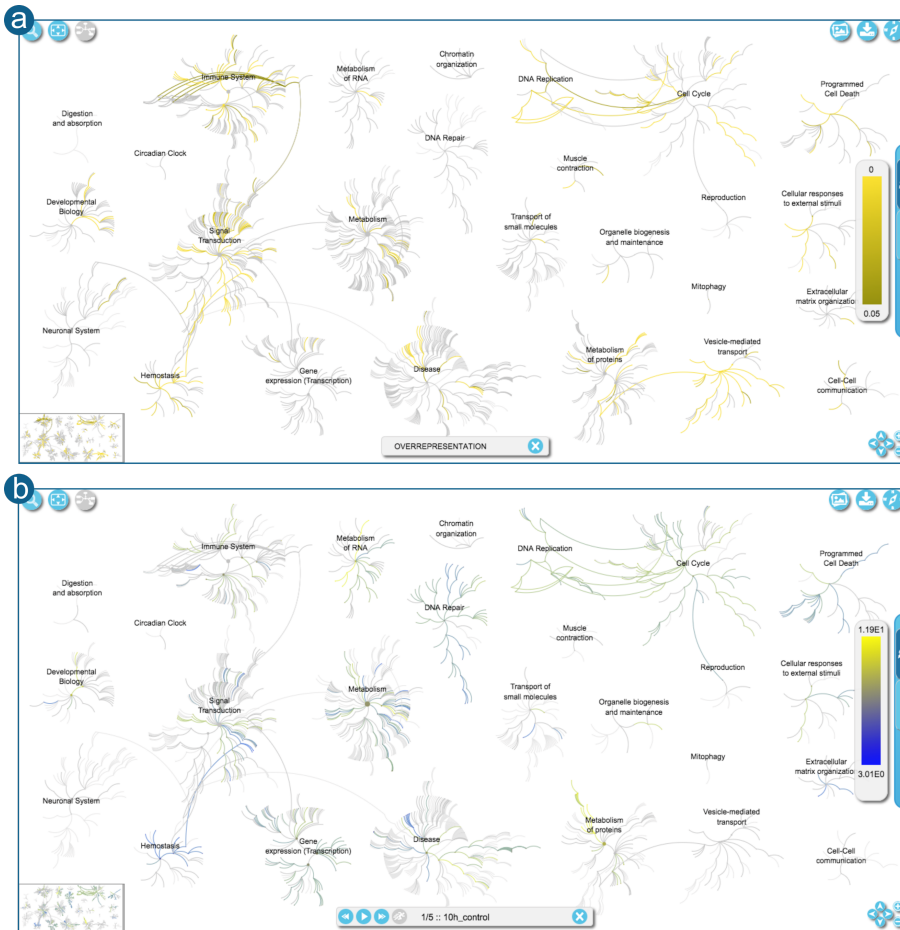


Figure 4.18. Different types of analysis overlay for the Pathways Overview widget. a) Overrepresentation analysis overlay. b) Expression analysis overlay.

When an analysis result is overlaid, the widget shows a control panel on the bottom centre of the viewport and a colour legend on the right-hand side (Figure 4.18). For enrichment and species comparison, the control panel identifies the analysis type and includes a button to remove the overlay (Figure 4.18 a). For expression analysis, the control panel also includes a mechanism to move through multiple samples, for instance a time series,

either manually by using the back/forward ‘double arrow’ buttons or automatically by clicking on the "play button" (Figure 4.18 b). Once playing is activated, the speed can be adjusted using the enabled ‘speed selection’ button. The expression analysis results overlay also takes into account the pathways significance.

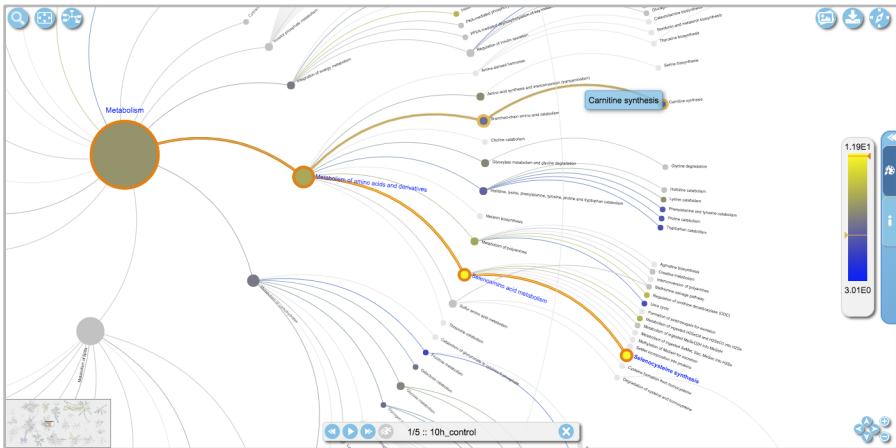


Figure 4.19. Pathways Overview zoomed in to “Metabolism” overlaying the results of an expression analysis. The node representing the “Selenocysteine synthesis” pathway is selected and the node representing the “Carnitine synthesis” pathway is hovered over with the mouse pointer. The legend on the right-hand side show the scale for the expression values overlaid and two flags appear; the one on top belongs to the selected pathway and the one below to the hovered over. These can be used to visually compare the expression values associated to the pathways.

For all types of analysis, the colours legend represents the results using colours based on the selected colour profile. When nodes are selected or hovered, the colour legend shows lines to indicate their significance or expression values (Figure 4.19). For enrichment and species comparison, if the hovered/selected pathway is not significant, the line does not appear.

The analysis result is also overlaid in the thumbnail and when a time series is "played" the thumbnail changes at the same time as the main view, so that when zoomed in in a specific area, it is still possible to observe changes across the entire pathway. Use of the "play" feature for a time series does not prevent the use of all the selection and movement features described previously.

4.3. SUMMARY

This chapter introduces a graphical pathway representation for use with Reactome data, enabling the representation of the complex parent-child relationships present in Reactome's hierarchical organisation. The main reason for having such a representation is to provide a means to overlay analysis result in such a way that the user can easily distinguish the most significant areas of biology represented in their data.

Although an existing force-directed layout algorithm was initially utilised, it did not achieve the expected results and a custom radial layout algorithm was developed instead. This was applied in two scenarios; (i) all pathways in one burst and (ii) each top-level-pathway in a different burst. The second scenario produced the desired result, so was used to fix the location of the nodes representing the top-level-pathway in a way that minimises the crossing of edges. The resulting custom radial layout algorithm only needs to be executed once for each Reactome quarterly release. Finally, a widget was developed to interactively render analysis results within the Pathways Overview. This widget has been part of the Reactome Pathway Browser <https://reactome.org/PathwayBrowser/> since release 52 [Fabregat et al. 2015].

Source code for the layout algorithm and widget is available in separate GitHub repositories. The layout is available at <https://github.com/reactome-pwp/fireworks-layout> and the widget at <https://github.com/reactome->

[pwp/fireworks](#). A JavaScript stand-alone wrapper was also developed and made available for third parties to reuse. Documentation for this widget is available at <https://reactome.org/dev/pathways-overview> and the code for the wrapper is at <https://github.com/reactome-pwp/fireworks-js>.

4.4. REFERENCES

Agarwal,P.K. and Erickson,J. (1998) Geometric range searching and its relatives. *Advances in Discrete and Computational Geometry, Contemporary Mathematics, American Mathematical Society, 23, 1-56.*

Bastian,M. Heymann,S. Jacomy,M. (2009) Gephi: an open source software for exploring and manipulating networks. In: *International AAAI Conference on Weblogs and Social Media, Association for the Advancement of Artificial Intelligence, San Jose, California.*

Bartels,R.H. Beatty,J.C. Barsky,B.A. (1998) "Bézier Curves." Ch. 10 in *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling. San Francisco, CA: Morgan Kaufmann, 211-245*

Lin,B. Chen,Y. Chen X. Yu,Y. (2012) Comparison between JSON and XML in applications based on AJAX. In: *International Conference On Computer Science and Service System, Nanjing, China, 11-13,1174-1177.*

Croft,D. O'Kelly,G. Wu,G. Haw,R. Gillespie,M. Matthews,L. et al. (2010) Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research, 39, D691-D697.*

Graham,J.W. (1999) Nicheworks - Interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics, 8(2), 190-212.*

Herrero,J. Muffato,M. Beal,K. Fitzgerald,S. Gordon,L. Pignatelli,M. et al. (2016) Ensembl comparative genomics resources. Database (Oxford).

Hughes-Hallett,D. Frazer,P. Gleason,A.M. Flath,D.E. Lovelock,D. Quinney,D. et al. (2013) Applied Calculus 5th Edition. John Wiley & Sons, Chapter 1.2 and 1.5.

Fabregat,A. Sidiropoulos,K. Viteri,G. Marin-Garcia,P. Ping,P. Stein,L. (2017) Reactome diagram viewer: data structures and strategies to boost performance. *Bioinformatics*, 34(7), 1208-1214.

Fabregat,A. Sidiropoulos,K. Garapati,P. Gillespie,M. Hausmann,K. Haw,R. et al. (2015) The Reactome pathway Knowledgebase. *Nucleic Acids Research*, 44, D481-D487.

Jacomy,M. Venturini,T. Heymann,S. Bastian,M. (2014) ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLoS ONE* 9(6), e98679.

Kobourov,S.G. (2012) Spring Embedders and Force-Directed Graph Drawing Algorithms, arXiv preprint arXiv: 1201.3011

Melançon,G. and Herman,I. (1998) Circular drawings of rooted trees. Reports of the Centre for Math. and Computer Sciences, Amsterdam, The Netherlands.

Nurseitov,N. Paulson,M. Reynolds,R. Izurieta,C. (2009) Comparison of JSON and XML Data Interchange Formats: A Case Study. *CAINE 2009*, 157-162.

Eades,P. (1992) Drawing Free Trees. *Bulletin of the Institute for Combinatorics and Its Applications*, 5, 10-36.

Petri,V. Jayaraman,P. Tutaj,M. Hayman,G.T. Smith,J.R. De Pons,J. (2014) The pathway ontology-updates and applications. *Journal of Biomedical Semantics*, 5, 7,

Teoh,S.T. Ma,K. Goodrich,M, Kobourov,S. (2002) Rings: A technique for visualization of large hierarchies, proceedings of the 10th International Symposium on Graph Drawing, London, UK, 268-275.

Wang,G. (2011) Improving data transmission in web applications via the translation between XML and JSON. In: *Third IEEE International Conference On Communications and Mobile Computing (CMC)*, Washington, DC, USA, 182-185.

5. DIAGRAM VIEWER

The Diagram Viewer is a Systems Biology Graphical Notation (SBGN)-based visualisation system that supports zooming, scrolling and event highlighting. It exploits PSI Common Query Interface (PSICQUIC) web services [Aranda et al. 2011] to overlay Reactome curated pathways with molecular interaction data from the Reactome Functional Interaction Network and external interaction databases such as IntAct [Orchard et al. 2014], BioGRID [Chatr-Aryamontri et al. 2017], ChEMBL [Gaulton et al. 2017] and MINT [Licata et al. 2012].

As demonstrated by their frequent usage in scientific literature and discussions, diagrams are important communication tools for biologists. From the user's perspective, representing pathway knowledge as a diagram is a helpful, intuitive way to represent information in a biological context and share it with others [Perini 2013].

The screenshot shows the Reactome Event Browser interface for the 'Cell Cycle, Mitotic' pathway. The top navigation bar includes links for About, TOC, User Guide, Data Model, Schema, Extended search, PathFinder, SkyPainter, Download, Linking, and Citing. The search bar contains the text 'everything' and 'with ALL of the words' in 'Homo sapiens'. The main content area displays a diagram of the cell cycle with phases G1, S-phase, G2-M, and Mitosis, and associated molecules like Cyclin D, Cyclin A, Cdk2, Wee1, Cdk5, Cyclin E, Cdk2, G0-G1, and G1-S. Below the diagram is a table with the following information:

Category	Value
Taxon	Homo sapiens
Represents GO biological process	mitotic cell cycle [GO:0000278]
Equivalent event(s) in other organism(s)	Cell Cycle, Mitotic [Mus musculus] Cell Cycle, Mitotic [Rattus norvegicus] Cell Cycle, Mitotic [Danio rerio] Cell Cycle, Mitotic [Fugu rubripes] Cell Cycle, Mitotic [Salix glauca]
Participating molecules	<ul style="list-style-type: none">26S protease regulatory subunit 4 [ytosin] UFG26S protease regulatory subunit 6A [ytosin] UFG

Figure 5.1. User interface for the first version of the Reactome Event Browser.

Reactome represents the steps of a pathway as a series of interconnected molecular events, known as “reactions”. Early implementations of the Reactome website did not have pathway diagrams, instead events were shown in a text-based format, sometimes with corresponding illustrations (Figure 5.1).

Diagrams were first made available in October 2010, when the first version of the Pathway Browser was released as a support for visual navigation and analysis of the Reactome data based upon SBGN [Croft et al. 2010]. Pathway diagrams graphically represent individual reaction events as molecular entities that are associated with a ‘reaction node’. By demonstrating that molecular entities are produced as reaction event outputs and subsequently used as the inputs of further events, reactions are connected in a graph. The resulting series of connected events represents a pathway.

Section 5.1 explains the state of the art of the Diagram Viewer, exposing both version 1 and 2 of this widget with their strengths and weaknesses. These two initial versions were developed by former Reactome staff members. Section 5.2 summarises two of the papers associated with this thesis describing the work that led to Diagram Viewer version 3.

5.1. EVOLUTION OF THE PREVIOUS VERSIONS OF THE DIAGRAM VIEWER

Reactome has always provided online tools to browse its content, but in terms of diagram visualisation and pathways analysis, it was October 2010 when the first web application to browse pathways in a SBGN-like format was launched [Croft et al. 2010]. SBGN is a standard that aims to specify the connectivity of biological graphs and the types of nodes and edges but does not specify their layout [Le Novere et al. 2009]. It includes methods to differentially

represent classes of molecule and multi-molecular entities such as complexes and includes several different classes of reaction-like event. The semantics of an SBGN diagram do not depend on the relative position of the symbols. While the labels of symbols are not regulated and only required to be unique within a map, it is important to note that SBGN does not depend on colours, patterns, shades and thickness of edges.

Le Novere et al. (2009) define three orthogonal, complementary types of diagrams for SBGN that can be seen as three alternative projections of the underlying more complex biological information; namely the process diagram, the entity relationship diagram and the SBGN activity flow diagram. Of the three types, the method selected as most appropriate for representing Reactome pathways is the entity relationship diagram because it emphasizes the influences that entities have upon each other's transformations, rather than the transformations themselves.

In the entity relationship diagrams used to represent Reactome curated pathways, each relationship represents a specific molecular event. The presence of entities in the map represents the effects they have upon each other and the sharing of entities demonstrates the connectivity between events. The annotated knowledge underlying this representation is manually extracted from experimental data in published scientific literature.

Reactome diagrams are closely aligned with the process of dividing Reactome content into a set of canonical pathways, which correspond to distinct biological processes, with minimal overlap of reactions, arranged in a hierarchy that corresponds to the Gene Ontology (GO) Biological Process hierarchy [GO Consortium 2000].

Pathway diagrams represent cellular compartments as boxes with a surrounding double line. A typical pathway diagram has a box representing the cytosol, bounded by a double-line that represents the plasma membrane. The white background outside the box represents the extracellular space. Other organelles are represented as additional labelled boxes within the cytosol. The icons representing molecular objects are placed onto the physiologically-correct box or associated boundary line to represent they are associated with the named cellular compartment or its containing membrane, respectively, e.g. a molecular object is placed on the boundary of the cytosol to represent that it is associated with the plasma membrane.

5.1.1. DIAGRAM VIEWER VERSION 1

At the time when the first pathway Diagram Viewer (Figure 5.2) was released, Reactome's content was organised into 161 canonical pathways, each represented and displayed in pathway diagrams following the SBGN standard. The Diagram Viewer offered interactive and dynamic pathway diagrams permitting zooming, scrolling and highlighting of events and molecules.

This first viewer version was developed entirely in JavaScript, which was chosen because the JavaScript engine was embedded in most popular Web browsers of the time (e.g., Internet Explorer, Firefox, Safari). As a result, no software installation was required, presenting a major benefit from the user's point of view.

Diagram content was displayed using a tiled maps approach, which involved creating the pathway images as a set of tiles on the server side during the release process, so that later they could be loaded on demand by the client. To enable different levels of zoom, each pathway diagram required several sets of tile images. To improve diagram loading time by reducing the amount of

data required, tiles were cropped to exclude retrieval of data not visible in the viewport, at the selected level of zoom.

Although graphically representing the pathway is the main purpose of the Diagram Viewer, another very important feature is allowing the user to interact and inspect the details of elements of the pathway such as reactions, reactants, products, catalysts or regulators. Selecting and/or hovering over elements are very common actions that need feedback to enrich the user experience. As the images themselves were not at this time interactive, a means to map the location of pathway elements onto raster coordinates was created to enable this requirement.

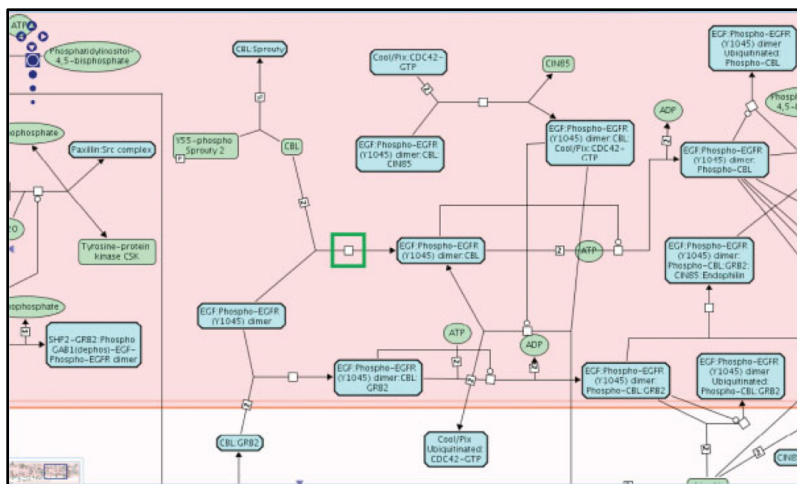


Figure 5.2. First version of the Diagram Viewer. The displayed pathway is “EGFR downregulation”. The selected reaction (green box) is “binding of CBL to EGFR” [Croft et al. 2010].

Displaying an image and superimposing visual feedback is a difficult task because the position of the elements in the viewer depends on the set of tiles that are loaded, which as explained above depends on panning and zooming

performed by the user. The solution adopted to meet this requirement was to place a green box around the selected element. An example of this is shown in Figure 5.2 where the green box is the result of selecting the reaction “binding of CBL to EGFR”. Diagram entities were also surrounded with a green box when selected.

The first version of the pathway Diagram Viewer did not offer a true zoomable user interface (ZUI), as the user could not pan or zoom the map in a satisfactorily fast interactive fashion [Latendresse et al. 2011]. The tiled maps approach was limited because both panning and zooming actions required server requests to retrieve the appropriate diagram tiles, as the content was shifted in the direction of pan actions or zoomed-in or -out resulting in a noticeable lag in response.

A major challenge for researchers and bioinformaticians is the integration of experimental and computational proteomics results with information relating to specific biological pathways [Haw et al. 2011]. One means to provide this interactivity for users is by implementing a system for data analysis and results visualisation (as described in Chapter 3). These analyses determine which pathways overlap significantly with a set of genes, proteins and/or compounds, specified by a submitted list of identifiers. Once a matching pathway is identified by the analysis, a useful means to indicate the matched entities in that pathway is to colour them. Overlaying analysis results is a very useful and popular feature that was included in the first version of the Diagram Viewer (Figure 5.3). The technique used was based on HTML floating DIVs placed on the viewport. Though the visual result was good enough to be usable, user interaction was slow and visually it was not appealing, particularly in the case of complexes and sets, where the overlay colour was

black and a second user action was required to visualise individual participating molecules.

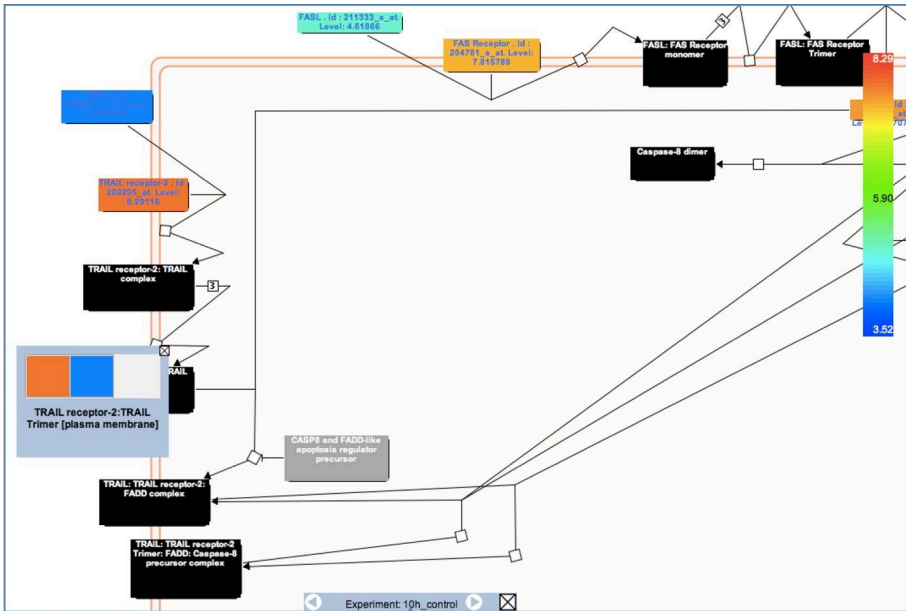


Figure 5.3. Diagram displaying the coloured physical entities that correspond to expression values of the experimental data. The nodes in this diagram are colour-coded: grey, no match; black, a (multicomponent) complex entity; and other colours represent expression levels. If the numerical data are a time series, the grey bar at the bottom of the coloured pathway diagram allows the user to step through time points and visualise the changes in the expression levels with the time of the individual genes involved in the pathway [Haw et al. 2011].

Although Reactome data sets are a high-quality resource for pathway-based data analysis, its usage as a platform for high-throughput data analysis is to some extent limited by its coverage of human proteins. Integrating molecular interactions (protein-protein and protein-compound) into Reactome pathway diagrams (Figure 5.4) was introduced as a way to extend coverage and give

access to functional annotations associated with the interactors [Haw et al. 2011].

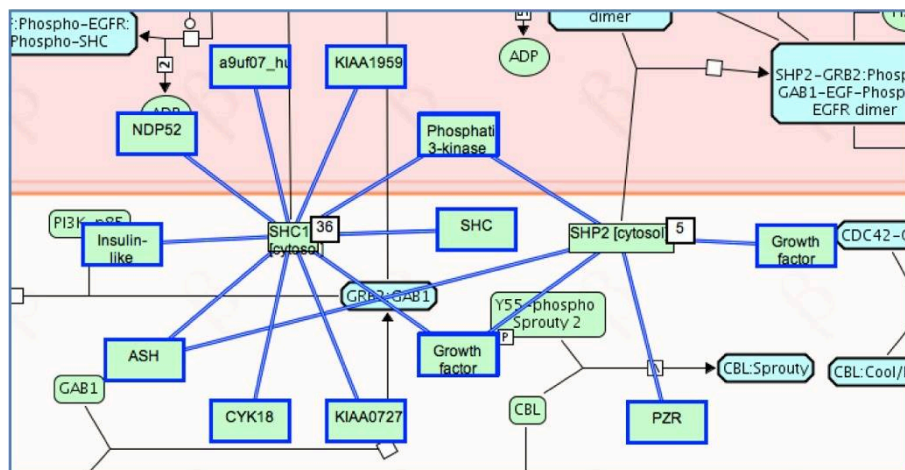


Figure 5.4. Protein–protein interactions displayed in the Pathway Browser for SHC1 [cytosol] and SHP2 [cytosol] physical entities [Haw et al. 2011].

Molecular interactions were integrated as an overlay that employed PSICQUIC web services to import binary interaction data from interaction databases (<https://github.com/PSICQUIC>). The default interaction database was IntAct. Two of the datasets, ‘Reactome’ and ‘Reactome-FIs’, were generated by the Reactome group. Molecular interactions were overlaid in the diagram interactively when selected by the user. One problem associated with this implementation was that there was no way to identify whether interactors were available prior to the user interaction. Consequently, it was necessary to click, wait and perhaps eventually find that no interactors were available. In addition, it was not possible to incorporate interactors as potential extensions of the pathway when performing pathway analysis, nor were interactors searchable within the pathway diagram.

Although this first version of the Diagram Viewer offered useful functionality, a number of improvements were suggested that could not be fulfilled due to limitations of the technology used. The most important of these was an improved visual feedback mechanism including a true ZUI, to improve the analysis results overlay and to provide a stand-alone widget that could be included in third party web pages. Achieving this required a new approach and selection of a new technology suitable for a long-term development and maintenance.

5.1.2. *DIAGRAM VIEWER VERSION 2*

To address the limitations encountered in the first version of the Diagram Viewer, a second version was proposed and developed for integration within the new Pathway Browser. To make integration with other elements of the Pathway Browser easier, the programming language chosen was Java, while GWT Toolkit was utilised for the framework. The technologies considered for graphic representation were HTML5 canvas and SVG. These two alternatives offer graphics features that are supported by major browsers. These technologies were, and still are, meant to address a range of different graphic scenarios.

Whilst HTML5 Canvas offers bitmaps with an immediate mode graphics API for drawing onto it, SVG offers a retained mode graphics model persisting in an in-memory model. SVG can be considered analogous to HTML because it builds an object model of elements, attributes and styles. HTML5 Canvas is a “fire and forget” model that renders its graphics directly to its bitmap and then subsequently has no sense of the shapes that were drawn; only the resulting bitmap persists.

While comparing HTML5 canvas and SVG, there was no obvious best approach. The decision required a comparison of the options before commitment to a single technology. The key consideration that led to rejection of SVG was that it results in slower rendering as document complexity increases due to its integration into the DOM. Consequently, the decision was made to implement the Diagram Viewer using HTML5 Canvas, despite its relatively poor text rendering capabilities and lack of animation, as it was determined that these deficiencies could be addressed by implementing layers for the final product.

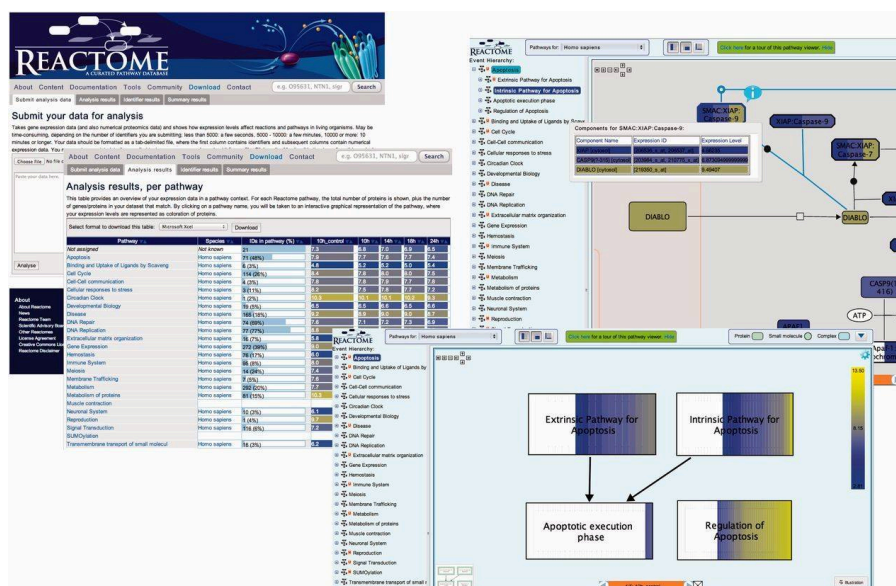


Figure 5.5. Pathway analysis overlay in the Diagram Viewer version 2. The data analysis workflow proceeds in three steps: data submission, a tabular display of results and visual display of results as an overlay in a Pathway Browser window. The fourth pane shows a detail of the Pathway Browser window, to highlight the display of expression levels [Croft et al. 2013].

The second version of the Diagram Viewer was released on October 2013. It included several improvements over its predecessor. The use of HTML5 canvas contributed to a better general look and feel. The previous tiled-map approach was replaced with a new approach that consisted of loading XML with the content and layout of the diagrams so these could be rendered using the HTML5 canvas API. This second version of the viewer improved the ZUI when compared with its predecessor by avoiding server requests for each user action of zooming or padding. It also provided an enhanced view of the analysis result overlay where the black filling for complexes and sets (Figure 5.3) was replaced with strips of varying size and colour, reflecting the expression values of contained elements (Figure 5.5).

In addition to the requirements of its predecessor, the second version of the Diagram Viewer included support for the visualisation of disease pathways. These can be split into two groups in terms of their development. The first group is infectious diseases, which differ from non-disease ('normal' pathways) only by the presence of microbial proteins or nucleic acids. The second group contains both gain- and loss-of-function events, commonly found in cancer and metabolic disorders, respectively. These events have an associated normal event counterpart, providing the framework that is used to represent the 'abnormal' event that occurs when the 'normal' event is disrupted by a mutation or similar disrupting molecular abnormality.

As explained previously, SBGN was the description language chosen to represent physical entities and their interactions in Reactome pathways. When disease pathways were included, the software had to be extended to create disease displays in which the modified entities and consequent event variations associated with a disease process are shown superimposed and

highlighted on top of the normal event in the pathway diagram (Figure 5.6) [Croft et al. 2013].

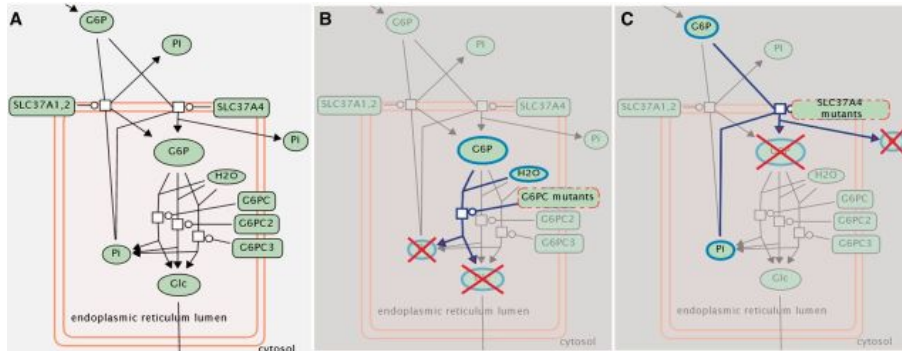


Figure 5.6. Example of disease curation and visualisation in Reactome. The normal process of glucose export from the liver under fasting conditions (A) is disrupted by mutations that block glucose-6-phosphate hydrolysis within the endoplasmic reticulum (B) or the transport of glucose 6-phosphate and orthophosphate (Pi) between the endoplasmic reticulum and the cytosol (C) [Croft et al. 2013].

A deeper analysis of the display details for this gain- and loss-of-function group of events reveals that they are rendered in a “three iteration strategy”. The first iteration renders the normal pathway without alterations or additions of any kind. The second iteration adds a grey layer covering the whole viewport where the diagram is being rendered. Finally, the third iteration overlays the changes caused by the disease.

5.2. DIAGRAM VIEWER VERSION 3

Systematic user experience testing and informal user feedback on the second version of the Diagram Viewer suggested that loading time and user interactivity needed improvement. These sessions revealed that users (i) had difficulty using and understanding the results of the diagram search functionality, (ii) found diagrams too crowded/complex, especially in

zoomed-out views, and (iii) often lost diagram context while navigating through the event hierarchy due to the slow loading of diagrams and abrupt changes of diagram graphic, rather than an animated transition to the target position. Other comments highlighted that users would prefer a progressive zoom rather than zooming in predefined steps.

Section 5.2.1 provides a summary of the paper “Reactome diagram viewer: Data structures and strategies to boost performance” [Fabregat et al. 2017] included as Publication P.2. This publication describes how problems identified in user experience sessions focused on version 2 of the Diagram Viewer were addressed in version 3. Sections 5.2.2 and 5.2.3 summarise the paper “Reactome enhanced pathway visualisation” [Sidiropoulos et al. 2017] (Publication P.3). This paper describes two new features included in version 3 of the Diagram Viewer. In this last paper, the author of this thesis appears last in the list of authors reflecting his role in the design and supervision of the work.

5.2.1. DATA STRUCTURES AND STRATEGIES TO BOOST PERFORMANCE

While requirement definition and project design of the work presented in this section were completed by the author of this thesis, code development was shared in equal parts with a colleague, Konstantinos Sidiropoulos. As mentioned previously, some of the data structures and strategies utilised in the development of version 3 of the Diagram Viewer were already included in the Pathways Overview widget.

The paper associated with this section describes how loading time and user experience were improved by implementing a more efficient diagram storage format and adopting new strategies for client data storage, retrieval and rendering. These improvements included: (i) a restructuring of the data format

used to send data from the server to the client, (ii) use of a graph data structure to store pathway content on the client side, (iii) boosting the client content load strategy, (iv) implementation of a multi-layer canvas approach, (v) utilisation of a space partitioning data structure to store the elements to be rendered and (vi) employment of a delegate design pattern to control the flow of information based on the level of zoom. This section describes each strategy in more detail.

Data format update

Version 2 of the Diagram Viewer used a custom eXtensible Markup Language (XML) format (<https://www.w3.org/TR/REC-xml>) for diagram data storage. JavaScript Object Notation (JSON) (<http://www.json.org>) is less verbose than XML and thus has a smaller footprint. Hence, the first step to improve the overall user experience was to reduce the client loading time by replacing the XML with JSON.

JSON's natural mapping to JavaScript objects is faster and uses fewer resources than its XML counterpart [Lin et al. 2012; Nurseitov et al. 2009; Wang 2011]. Therefore, for version 3 of the Diagram Viewer all Reactome pathways containing diagram layout information were converted from XML to JSON and stored on the server side as static resources during the release process. The resulting files are smaller in size than the equivalent XML file (Figure 5.7a), which improves processing time (Figure 5.7b).

As part of this process, a graph of all the entities and reactions contained in the pathway diagram is generated and stored in an additional JSON file. This enables a richer browsing and search experience throughout the diagram content since the new graph file contains information for all the pathway

participants along with its structures. The next subsection elaborates on the creation of the graph and its usage along with the layout information.

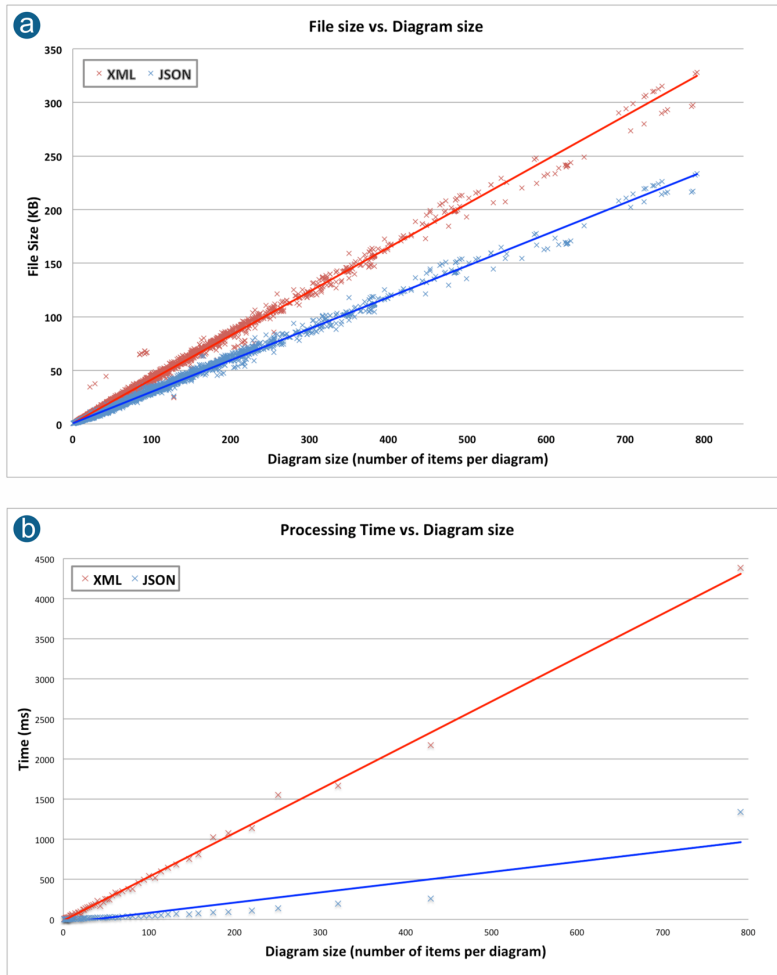


Figure 5.7. XML vs JSON comparison charts. (a) Comparison of file sizes for XML and JSON formats versus the diagram size in terms of number of entities present per diagram. (b) Comparison of processing times achieved by Diagram Viewer v2.0 (consuming diagrams in XML) and Diagram Viewer v3.0 (consuming diagrams in JSON) versus the total number of diagram entities. Measured over all human pathway diagrams present in Reactome data version 52 [Fabregat et al. 2017].

Underlying graph structure

Amongst other elements, diagrams contain macromolecular complexes and entity sets, comprised of components and members, respectively. Entity sets are used to group entities that have a common property and are interchangeable. Complexes are the result of several molecules coming together to create a single functional entity. Sets and complexes may contain complexes or sets as their constituents [D'Eustachio 2011].

This aggregating and nested approach can quickly generate highly structured networks of entities that can be represented in a diagram by a single glyph, greatly simplifying the view. However, the Diagram Viewer needs to both present this simplified view while allowing access to the full complexity of the represented data. In particular, the Diagram Viewer needs to include a search function that is able to find the constituents of complexes and sets that may not be visible in the diagram and to allow the user to discover the constituents of complexes and sets.

In previous versions of the Diagram Viewer, the client retrieved a file with the identifiers defining each element present in the diagram from the server side. In the new version, a file with a graph representing the content of the different complexes and sets for each diagram and annotating the participants of every included reaction is required (Figure 5.8). This approach introduced an additional file with the graph content that is consumed separately by the client and merged with the layout data, once both are loaded.

The graph and layout content have elements in common but in most cases the graph contains more information. In the example presented in Figure 5.8, the pathway diagram layout contains 7 elements; 5 entities and 2 reactions (Figure 5.8a), and the graph contains 11 elements; 9 entities and 2 reactions (Figure

5.8b). The 4 extra elements in the graph represent components of C_2 (4 entities) that are not present as individual entities in the layout. A less obvious benefit of the graph is that entities that are part of different complexes or sets are represented only once but remain accessible via graph traversing.

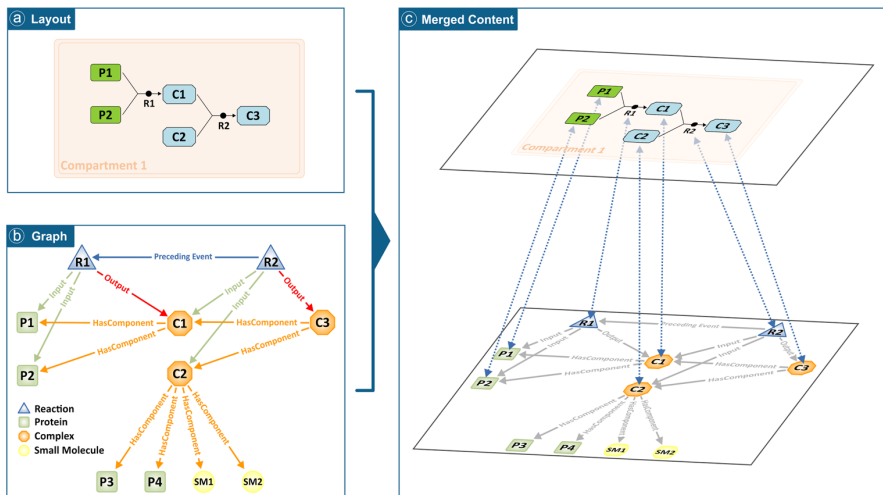


Figure 5.8. Schematic view of a pathway made up of two reactions. (a) The pathway diagram as presented to the final user. (b) Underlying graph with the whole content of the pathway. (c) Representation of the merging of both the diagram and graph on the client side. In the figure, P_n are proteins, SM_n are chemicals, C_n are complexes and R_n are reactions. From the graph, it can be extracted that C_1 contains $[P_1, P_2]$, C_2 contains $[P_3, P_4, SM_1, SM_2]$ and C_3 contains $[C_1, C_2]$, but by traversing the graph it can be easily inferred that C_3 actually contains $[P_1, P_2, P_3, P_4, SM_1, SM_2]$ [Fabregat et al. 2017].

The information stored in the layout and graph files is complimentary; the client side code implements a technique that merges the information from them (Figure 5.8c). The approach used is to propagate user actions from the layout level to the graph level, creating an intuitive means to traverse diagram content and allowing the highlighting of relevant entities by traversing from

the graph to the layout. In addition, the diagram search now has the ability to find both entities that are represented by a glyph in the diagram and contained entities within the set or complex represented by that glyph. This allows the user to search all the entities and components/members of complexes/sets present in a single diagram. The client can highlight all the diagram entities that match the search term.

Updated loading and caching strategies

The introduction of separate layout and graph files was accompanied by the adoption of a render-first client loading strategy (Figure 5.9). The client makes concurrent XMLHttpRequest calls for the layout and graph data (<https://xhr.spec.whatwg.org>). Once the layout data is available, the viewer processes it and renders the diagram on the canvas. When the graph content is ready, it is processed and linked to the diagram layout and used for interactive navigation, search and if performed, analysis overlay purposes. This render-first approach primes the Diagram Viewer with the layout while it retrieves the graph behind the scenes. This strategy improves user experience by reducing both the true and perceived loading time.

Strategies to prioritise the loading of information that is sufficient to render something on the screen and engage the user are particularly useful when full visualisation may require an extended loading period. People can only define a duration if there are clear start and end times [Seow 2008]. Consequently, when a user can see something rendered on the screen they assume loading has ended. Full loading can continue behind the scenes.

Users often go back and forth between pathways, causing the viewer to load and show the same diagram several times. Therefore, a pathway diagram that has been viewed is very likely to be revisited again shortly afterwards. In

computer science, this is known as locality of reference [Denning 2005] and is a clear use case for cache mechanisms. The Diagram Viewer implements a Least Recently Used (LRU) caching mechanism [Denning 1968] to record the layout and view status (zoom level and panning) of the most recently viewed diagrams so that when a diagram is revisited, the viewer does not need to request data from the server, instead it uses cached data to display content as it was previously viewed.

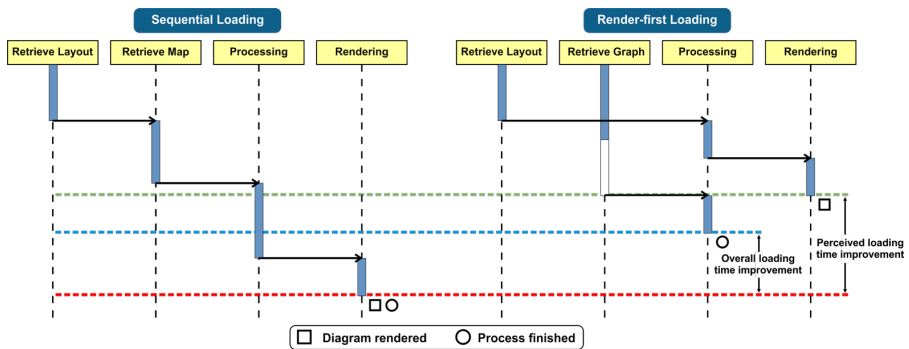


Figure 5.9. UML sequence diagram comparing sequential and render-first loading strategies. The difference between the blue and red lines shows the true loading time improvement. The improvement in the perceived loading time is highlighted by the difference between the green and red lines [Fabregat et al. 2017].

Multi-layer HTML5 canvas strategy

The new version of the Diagram Viewer responds to common user actions, such as hovering over an element with the mouse or selecting an entity in the diagram, by highlighting hovered elements and marking the selected entity, respectively. To improve user experience and visually reinforce user actions, the Diagram Viewer draws a halo around selected elements (reactions and participating entities). In addition, when the user selects an entity that is repeated in the same diagram, the viewer marks all instances of that entity as selected and draws halos around all elements related to them.

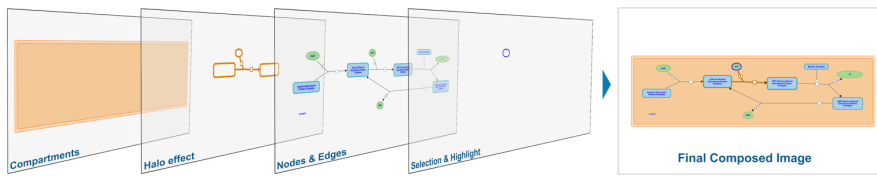


Figure 5.10. A simplified example of the multi-layer canvas strategy. The first four images from left to right represent different layers that compose the final image: (1) Cellular compartments, (2) Halo effect, (3) Nodes and Edges and (4) Selection and Highlight. The rightmost image shows the pathway diagram as seen from the user’s perspective [Fabregat et al. 2017].

To improve visual feedback and optimize the diagram rendering process, the new version of the viewer implements a set of advanced techniques used in the gaming industry. In particular, the multi-layer canvas approach was adopted to reduce the processing and redrawing overhead inherent to a single canvas update when the user interacts with it or animation is performed (<https://www.ibm.com/developerworks/library/wa-canvashtml5layering>).

Each of the stacked canvases in Figure 5.10 represents a conceptual layer and is reserved for drawing specific types of glyphs corresponding to different diagram objects such as compartments, reactions, nodes, entities or interactors. By employing this technique, only those layers that require redrawing are updated, resulting in reduced rendering times for actions such as highlighting or selection. This enhances the user experience.

For instance, while the user moves the mouse pointer across a diagram, only the ‘Selection and Highlighting’ layer needs updating in order to reflect the changes in the highlighted element. Similarly, when a diagram element is selected, only the ‘Halo effect’ and ‘Selection and Highlight’ layers require updating.

Space partitioning data structure

Identifying the elements under the mouse pointer is a computationally demanding task if it is performed by a brute force or exhaustive search algorithm [Knuth 1997]. The cost of an exhaustive search algorithm is a linear function of the number of elements to be searched, $O(n)$ in big O notation. Determining whether the mouse pointer position intersects with the area each element occupies can be slow, delaying the action of highlighting and making the interface appear unresponsive to the user.

To address this problem a QuadTree was used in the new implementation. A QuadTree is a tree data structure used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions [Finkel and Bentley, 1974]. The QuadTree is employed to efficiently (i) query only those diagram entities present in the viewport that need to be rendered and (ii) identify entities when hovered over, or selected without the need to use the brute force method.

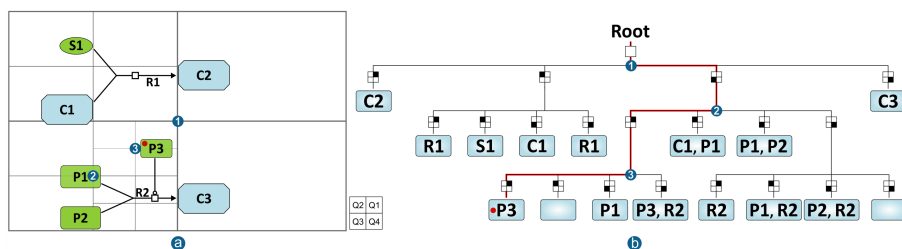


Figure 5.11. Hypothetical diagram composed of two separate reactions where (a) shows how the viewport is recursively split into different quadrants, so each of them contains two or less elements, (b) is the representation of the resulting QuadTree to achieve the two-dimensional space partitioning. The red dot in (a) represents the mouse pointer location and the red path in (b) depicts the tree traversing steps to narrow down the elements to be checked against the mouse location [Fabregat et al. 2017].

Figure 5.11 provides an example of how elements in a diagram are located in a QuadTree with quadrant size 2, meaning that only two objects are allowed per quadrant. The red line in Figure 5.11b highlights the path traversed in the tree to identify the element under the mouse pointer (red dot) based on a series of quick comparisons between the mouse coordinates and every quadrant centre starting for the root and progressively moving down the nodes of the tree. From the root (centre of the viewport) the red dot (Fig. 5.11a) is the 3rd quadrant (Q3); from the centre of Q3 the red dot is in the first quadrant (Q1); from the centre of Q1 the red dot is again in its first quadrant (Q1). Since this last quadrant is not further split, the position of the mouse pointer only needs to be compared against the contents of that quadrant, which in this case is only P3. Thus, determining that P3 is the element hovered over by the mouse pointer takes three quadrant comparisons and checking only one element of the nine present in the diagram. This provides a significant improvement over the brute force method that would check the mouse position against every element present in the diagram.

For the new Diagram Viewer, the QuadTree was extended to work not only with points but also with shapes that occupy diagram areas, thereby further narrowing the number of elements to be redrawn by determining whether they are in the part of the diagram visible in the client viewport. This allows a fast, selective redraw limited to visible regions of the diagram, improving interactivity of the Diagram Viewer.

Renderer delegates

To meet users' requests for less cluttered pathway diagrams but at the same time preserve access to all the information stored in Reactome, the new viewer controls the amount of visualised information by zoom level. In practice, this means that the viewer enriches or abstracts layers of information, depending

on the zoom level. Thus, diagram entities are rendered according to the level of zoom, progressively revealing more details as the user zooms in (Figure 5.12). For instance, common ‘housekeeping’ molecules, such as ADP, ATP or water are hidden in the zoomed-out view, resulting in less crowded diagrams.

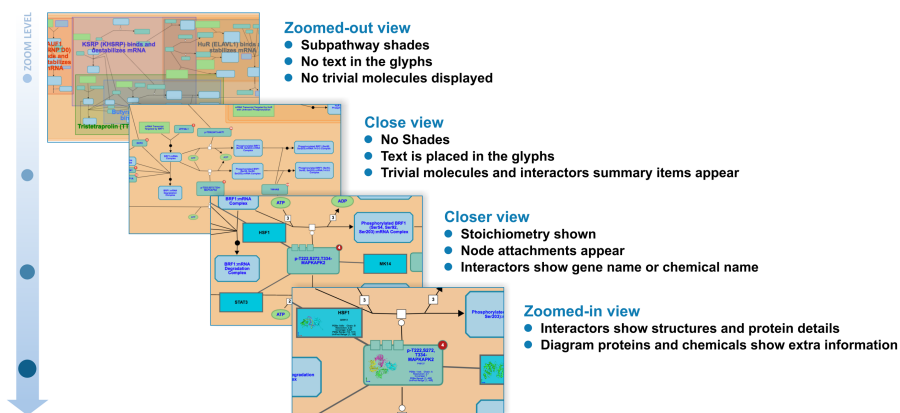


Figure 5.12. Diagram Viewer version 3 controls the flow of displayed information by the level of zoom abstracting or enriching the view with layers of information [Fabregat et al. 2017].

This strategy also improves rendering time because fewer details are drawn. Other simplifications include not rounding off of box corners, showing reaction backbones without the central reaction node and removing node attachments and boxes to indicate stoichiometry. As the user zooms in to a specific area the number of displayed elements falls and more detail is added.

5.2.2. TEXT BOOK STYLE ILLUSTRATIONS: ENHANCED HIGH LEVEL DIAGRAMS

The pathway diagrams interactive viewer has a limitation that makes it less well suited to the display of pathways at higher levels of the hierarchical organisation. Higher level pathways are aggregators that represent biological

functions or parts of the same process. It is often impossible to represent molecular details of the entire process in a single, readable diagram. The paper associated with this section [Sidiropoulos et al. 2017] presents recent updates in the Reactome web interface that improve visualisation and navigation of Reactome pathways, particularly at higher levels of the hierarchy, as well as new options for downloading and re-using the pathway diagrams.

As shown in Figure 5.13, higher level pathway diagrams previously did not represent each molecular event, instead entire subpathways were represented as a single glyph consisting of a box with a thick green boundary that served as a link to lower levels of the hierarchy. This allowed the user to navigate to the lower subpathway levels with their associated detailed pathway diagrams but it offered no visual feedback on the contents or relationships between subpathways or any notion of sequence or dependence.

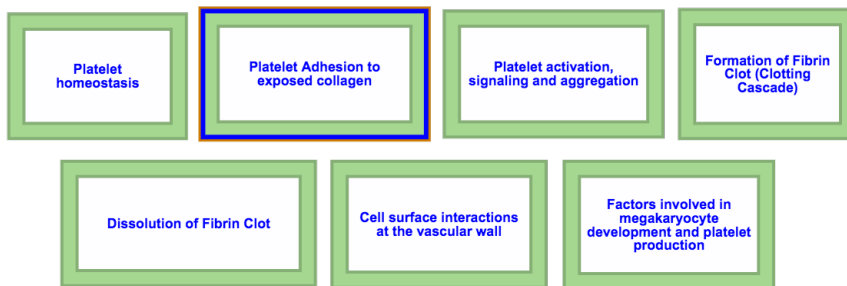


Figure 5.13. Hemostasis as previously represented with subpathway icons.

User experience testing identified that users struggled to identify the process represented in these type of Reactome pathway diagrams. This was at least partly a consequence of the use of the SBGN standard, which despite being a recognised standard in the systems biology community, is unfamiliar to many biologists. Illustrations are an established means to help people understand the relationships between and within processes.

Vision is the dominant sense in Humans [Schneck 1996], so illustrations can make a pathway easier to comprehend, but the styles used in textbooks and journals vary enormously. Static figures and illustrations were incorporated into the Reactome website from the earliest version but not consistently and with a variety of styles.

In the earliest implementation, Reactome offered static illustrations that were displayed inline with other details, later these were linked as a selectable side resource (Figure 5.14). Although these static illustrations were well received, they lacked interactivity and could not be automatically enriched with overlays or visual feedback or linked to the corresponding pathway diagram.

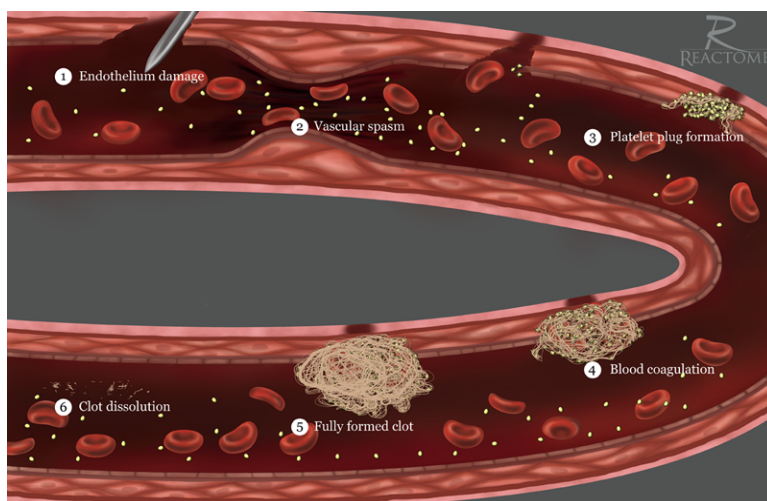


Figure 5.14. Hemostasis as an Illustration.

Feedback on the small number of available illustrations suggested that those would be well-received by Reactome users, particularly as a replacement for the higher-level diagrams that consisted entirely of subpathway icons. It was also clear that it would be beneficial to use a set of standardised glyphs in

these illustrations that make use of a common iconography to help the user understand the represented process.

To improve upon these illustrations and make them interactive in the scope of Reactome's Pathway Browser, a number of requirements had to be considered. These requirements can be split between technical and conceptual. Technically, the illustration had to be produced in a format that could be easily processed by a computer, for example SVG. Conceptually, the illustration (i) had to have a one to one mapping between what was annotated (present in the represented level of the hierarchy) and its sub levels and (ii) the main elements contained in the new format needed to be annotated and stored within SVG in a manner that makes object types recognisable for the software that displays them (explained in detail on this page of Reactome documentation <https://reactome.org/icon-info>). To achieve this, Reactome recruited an expert graphic designer to produce the illustrations, referred to internally as Enhanced High Level Diagrams (EHLDs).

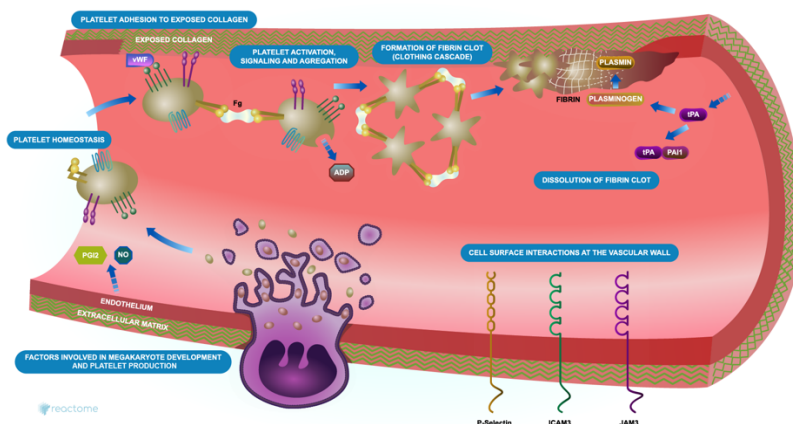


Figure 5.15. Hemostasis as an Enhanced High Level Diagram (with the SVG).

The designer used sketches produced by Reactome curators to create EHLDS that meet the conceptual requirements, producing SVGs that comply with the technical requirements. An example is shown in Figure 5.15. When compared with Figure 5.13, the pathway diagram is no longer a set of subpathway boxes, instead it combines the newly-designed illustration logic with a one to one mapping between areas in the diagram and sub-pathways. The EHL is “interpreted” by the software and is fully interactive, offering all the features available in detailed pathway diagrams such as hovering over elements, selection, flagging and importantly, pathway analysis results overlay.

EHLDS, including analysis results overlays if present, can be downloaded and saved in SVG format. The exported files can be edited in commercial and open-source graphics applications. EHLDS use a consistent iconography that reuses glyphs when the represented entity has a role in more than one biological process. For example, all representations of platelets in the Hemostasis EHL use same symbol. A library of these graphical elements in SVG, PNG and EMF formats is available at <https://reactome.org/icon-lib>, distributed under the terms of the CC-BY license (<https://creativecommons.org/licenses/by/4.0/>). The aims of providing such a library are to facilitate the creation of uniform diagrams through the use of pre-existing glyphs and to offer these components to the community for reuse.

The work presented in this Section has been performed by Reactome staff under the direct supervision of the author of this thesis. More specifically, code written to support the use of EHLDS in the Diagram Viewer was developed by Konstantinos while the illustrations were created by Christoffer Sevilla with the help of all Reactome curators, but particularly Steve Jupe. At the time of writing, the available Reactome database was version 63 where

80% of diagrams previously represented entirely by subpathway boxes have been replaced by EHLs.

5.2.3. *OVERLAYING ANALYSIS RESULTS*

The Diagram Viewer has a module that integrates the Reactome Analysis Service. Analysing data produces an analysis token (Section 3.2.1) that can be set so the result is displayed as an overlay on top of both regular diagram and EHLs [Fabregat et al. 2015 and 2017]. For regular diagrams, the analysis overlay feature was available in version 1 and 2.

Version 3 improves on the functionality of version 2 by enhancing the method used to produce the analysis result overlay for regular pathway diagrams. The analysis overlay is implemented at the level of the renderer delegates and also takes advantage of the underlying graph structure, both described in the previous subsection.

Following over-representation analysis (Figure 5.16a), pathway diagram entities that were represented in the submitted data set are re-coloured, using yellow in the default colour scheme. Complexes, sets and subpathway icons are coloured to represent the proportion of molecules contained by the entity that were represented in the submitted identifier list. In Figure 5.16a, PRKCA is yellow indicating that it was in the submitted list. CREB1 was not in the submitted list so it is not re-coloured. The complex of “Protein Kinase C, alpha type: DAG” is part re-coloured, part not, indicating that some molecules in the complex were represented in the submitted dataset.

Following expression data submission, objects in the diagram are re-coloured according to the numeric values submitted with the identifiers (Figure 5.16b). The colours used represent expression values corresponding to the scale presented as a bar on the right-hand side. There are several colour schemes,

Entities that were not represented in the input data are not re-coloured. Entities with bands of colour represent complexes or sets containing more than one molecule. When zoomed out, the colour of the band reflects the average of the values submitted, for molecules represented in the dataset. The size of the band reflects the proportion of molecules that had submitted values. At higher levels of zoom each molecule is represented by a band, coloured according to the associated submitted value, arranged alphabetically by name. If multiple columns of values were submitted, representing multiple samples, e.g. time-points or a disease progression, the order of the bands is the same for each sample.

When expression analysis is overlaid, the experiment browser toolbar (bottom of Figure 5.16b) is used to step through overlays representing multiple columns of data. The intended usage is with, for example, a time-series or disease progression. The user can move between them by clicking the arrow buttons or the play feature. The header of the data column (if present) is displayed between the arrows. The Pathway Diagram re-colours to reflect the new values.

The viewer allows users to overlay pathway analysis results onto EHLs (Figure 5.17a). The results are displayed in the label of each subpathway. The label is overlaid by a coloured rectangular shape; its width and colour represent the percentage of hit entities and the P-value, respectively (Figure 5.17b). Subpathways with a P-value below a certain threshold ($P < 0.05$) are coloured in grey. For hit subpathways, additional information about the hit elements and the false discovery rate are displayed next to the label. Upon hovering or selection of a subpathway, its P-value is indicated in the coloured legend bar displayed on the right side of the viewport. As explained in detail in Chapter 3, analysis results are temporarily stored on the Reactome server.

Office tools such as Microsoft PowerPoint, Apple Note, or Apache OpenOffice Impress allow the creation of multimedia presentations where the users can rearrange the contained objects and customise other properties such as size, colour and shape. Providing the ability to export Reactome diagrams in the standard format used by these tools (PPTX) allows users to benefit from these features. This may appear to be a simple export process but there were several issues to address.

Reactome diagrams use SBGN to represent their content, but the storage format was defined ‘in-house’. The latter was originally thought to improve raster-representation based algorithms. Interconnections between different objects, as well as anchor points between reactions shapes and their participants, were not defined. To achieve the desired result when exporting the diagram as objects to a document (Figure 5.18), this limitation had to be programmatically addressed during the conversion phase.

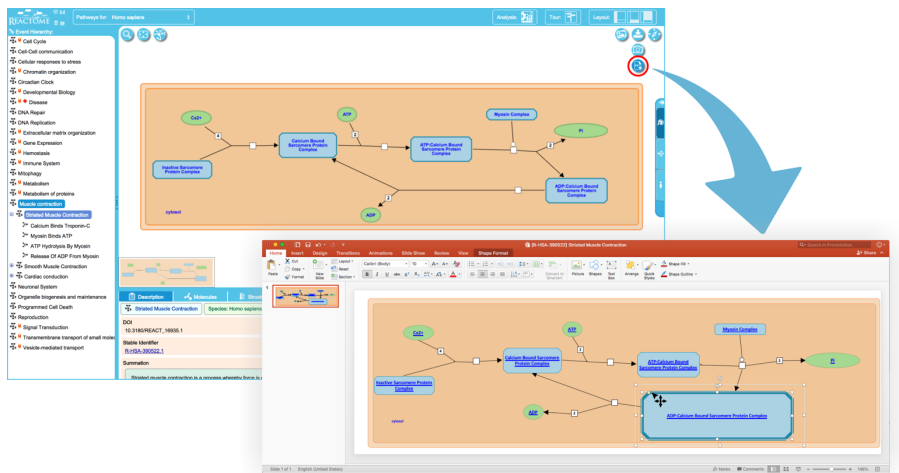


Figure 5.18. The pathway diagram for Striated Muscle Contraction exported to Microsoft PowerPoint (<http://reactome.org/PathwayBrowser/#/R-HSA-390522>) [Sidiropoulos et al. 2017].

In addition, different techniques were used in pathway diagrams for the various objects that represent a reaction and its associated relationships with other reactions or regulators, such as invisible anchor points to join segments of reactions and grouping of objects that visually define reactions properties (i.e. catalyst and regulatory line endings). Each required a solution to permit representation in the exported document. Additionally, the PPTX exporter was designed to reflect the user's preference for colour profile and other display options available in the user interface.

PowerPoint exporter code was developed by Guilherme Viteri (Reactome developer) under the direct supervision of the author of this thesis. Although the design and definition of requirements was performed by the author of this thesis, Guilherme played a key role in the evaluation of the different libraries to generate the Power Point files.

5.2.5. RESULTS AND DISCUSSION

The new pathway Diagram Viewer combines the strategies and data structures described above to improve performance and includes new features that aim to address shortcomings in the previous version as highlighted by user experience testing. An updated diagram storage format combined with an improved 'render-first' loading strategy has resulted in faster loading of diagrams.

Additionally, faster rendering was accomplished by a combination of (i) a QuadTree to identify the elements required to draw the visible area, (ii) rendering delegates that declutter the view by regulating the level of detail to be drawn and (iii) a multi-layer HTML5 canvas strategy that optimizes rendering by updating only those layers that require redrawing. Optimised rendering enables the introduction of animation and smooth transitions that

help users maintain an understanding of the diagram context while navigating through pathways. The underlying graph structure provides the basis for an improved built-in search feature that can consider all the molecules participating in the pathway, whether they are visible or not.

Updating the storage format had a positive impact on performance. To assess this, a comparison was done for the file sizes of the previous (XML) and new data format (JSON) and the time required by the client to process them, including the time required to populate the model in the client side with diagram data once retrieved.

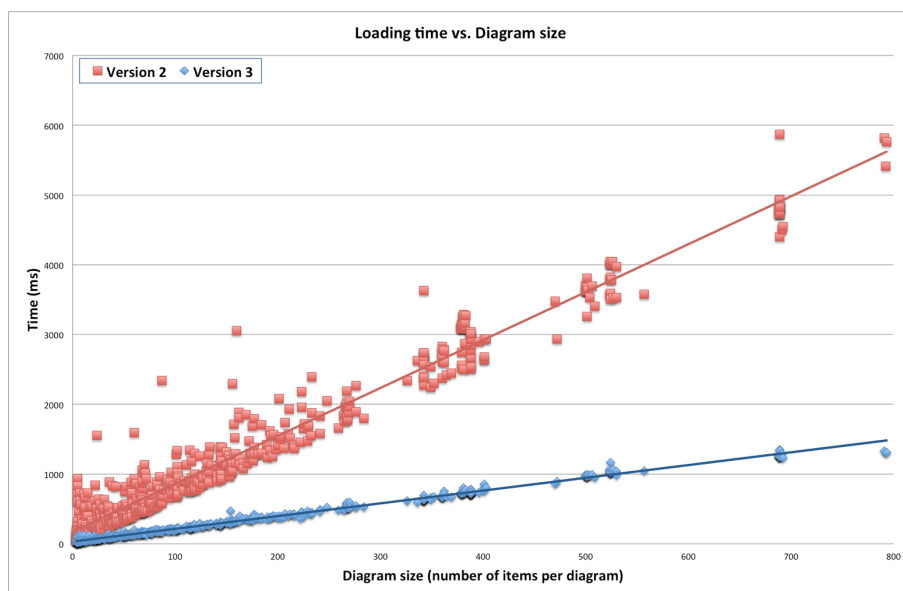


Figure 5.19. Comparison of perceived loading times achieved by Diagram Viewer version 2 and version 3 versus the diagram size (in number of diagram entities). Measured over all human pathway diagrams from Reactome data version 52 [Fabregat et al. 2017].

The update in the storage format combined with the new render first loading strategy contributed to reducing the overall diagram loading time, as it is perceived by the user. This includes the time required until the diagram is loaded and fully rendered by the client. Figure 5.19 presents a chart comparing the times required by the previous and the new version of the client to display diagrams against the diagram size (measured in number of entities present in a diagram).

A striking feature of the comparison of perceived loading times (Figure 5.19) is that the new Diagram Viewer is both faster and more consistent. Loading times measured for the previous Diagram Viewer exhibit high variability, especially for smaller diagrams, because the number of items drawn in the diagram does not represent the size of the pathway in terms of participating molecules. Complexes and sets contain multiple participant molecules; encapsulated pathways will also add a large number of participants. Pathways at higher levels of the Reactome hierarchy are frequently represented as diagrams that contain only subpathways, it is unsurprising that they contain a large number of participants. Combined with the inefficient sequential loading strategy (Figure 5.9), this explains why relatively small pathways, with only a few entities, required up to 1.5 s to load. Simply put, before rendering anything on screen, the previous client had to retrieve and parse a large amount of information to create a map of all participating entities.

As illustrated in Figure 5.19, the new version of the pathway Diagram Viewer achieves better performance for all Reactome diagrams. The new version of the client accomplishes loading and rendering of 97% of Reactome diagrams in less than 1 second (versus 57% previously); 74% of the total number of diagrams are loaded and rendered in under 0.5 s (versus 31% previously). This has a positive impact on the user experience and is particularly important in a

web application that runs in a web browser environment, where code is executed in a single thread, without use of concurrency. The adoption of a multi-layer HTML5 canvas strategy and space partitioning data structure helps minimize CPU workload and allows new features such as animated transitions without penalising the user experience.

EHLs have been designed to contain images that will be familiar to biologists. They use a consistent iconography that is based on a survey of typical textbook and online representations of the process. The intention is to make the navigation experience more intuitive and visually pleasing; the user will recognize the process that is represented and be able to select the appropriate region of the illustration to navigate to the next level of the Reactome hierarchy level without the need to read and understand text labels; ultimately the user will arrive at a classic, detailed pathway diagram that represents the molecular mechanism underlying the pathway. For users who are not familiar with the graphical representation used in EHLs the text labels are retained.

5.3. SUMMARY

Through the use of highly optimised data structures and algorithms, Reactome now has a pathway Diagram Viewer that is improved in terms of performance and usability. The new version provides a robust, scalable pathway visualisation that is easily integrated into third party applications. It was developed to be both extended and easily integrated in third party applications. Reactome currently offers two options for integrating the pathway Diagram Viewer in other web applications using either the GWT implementation or the JavaScript wrapper. Details and examples can be found at <https://reactome.org/dev/diagram/>.

The new features presented above are fully integrated in the current version of the Reactome Pathway Portal. EHLDs and the option to export diagrams to PowerPoint are available to users via the Pathway Portal and through the stand-alone version of the Diagram Viewer widget, which is available for integration in third party applications (<http://reactome.org/dev/diagram>).

With the collaboration of Reactome curators and designers, informative and visually appealing EHLID illustrations have been created. Feedback from users regarding this new feature has been very positive. In particular, users found the new, interactive graphical representation of Reactome pathways more descriptive of the biological process and consequently more intuitive to navigate when compared to the previous static diagrams.

Visual appeal and navigability issues of previous Reactome diagrams have been addressed by the introduction of EHLIDs, which represent biological processes in a familiar textbook style that allows intuitive navigation to more specific subtopics. In addition, EHLIDs are used to represent summarised analysis results. EHLID images and associated analysis result overlays can also be exported in a lossless, editable format, enabling users to represent their own research results in the context of Reactome pathway diagrams. In addition, the Reactome pathway iconography library provides graphical representations of common molecular biology elements suitable for use in slides and publications. Finally, classic pathway diagrams can now be exported in PPTX format, allowing their editing and reuse with familiar presentation software.

5.4. REFERENCES

Aranda,B. Blankenburg,H. Kerrien,S. Brinkman,F.S. Ceol,A. Chautard,E. et al. (2011) PSICQUIC and PSIScore: accessing and scoring molecular interactions. *Nature Methods*, 8, 528-529.

Chatr-Aryamontri,A. Oughtred,R. Boucher,L. Rust,J. Chang,C. Kolas,N.K. et al. (2017) The BioGRID interaction database: 2017 update. *Nucleic Acids Research*, 45,D369-D379.

Croft,D. Fabregat,A. Haw,R. Milacic,M. Weiser,J. Wu,G. et al. (2013) The Reactome Pathway Knowledgebase. *Nucleic Acids Research*, 42, D472-D477.

Croft,D. O'Kelly,G. Wu,G. Haw,R. Gillespie,M. Matthews,L. et al. (2010) Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39, D691-D697.

Denning,P.J. (2005) The locality principle. *Communications of the ACM*, 48, 19-24.

Denning,P.J. (1968) The working set model for program behavior. *Communications of the ACM*, 11, 323-333.

D'eustachio,P. (2011) Reactome knowledgebase of human biological pathways and processes. *Methods in Molecular Biology*, 694, 49-61.

Fabregat,A. Sidiropoulos,K. Viteri,G. Marin-Garcia,P. Ping,P. Stein,L. (2017) Reactome diagram viewer: data structures and strategies to boost performance. *Bioinformatics*, 34(7), 1208-1214.

- Fabregat,A. Sidiropoulos,K. Garapati,P. Gillespie,M. Hausmann,K. Haw,R. et al. (2015) The Reactome pathway Knowledgebase. *Nucleic Acids Research*, 44, D481-D487.
- Finkel,F. and Bentley,J.L. (1974). Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4, 1-9.
- Gaulton,A. Hersey,A. Nowotka,M. Bento,A.P. Chambers,J. Mendez,D. et al. (2017) The ChEMBL database in 2017. *Nucleic Acids Research*, 45(1), D945-D954.
- GO Consortium (2000) Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25, 25-29.
- Haw,R. Hermjakob,H. D'eustachio,P. Stein,L. (2011) Reactome Pathway Analysis to Enrich Biological Discovery in Proteomics Data Sets. *Proteomics*, 18, 3598-3613.
- Knuth,D. (1997) *The Art of Computer Programming. 3: Sorting and Searching*. 3rd edn. Addison-Wesley, Reading, MA, 396-408.
- Latendresse,M. and Karp,P.D. (2011) Web-Based Metabolic Network Visualization With a Zooming User Interface. *BMC Bioinformatics* 12, 176.
- Le Novère,N. Hucka,M. Mi,H. Moodie,S. Schreiber,F. Sorokin,A. et al. (2009) The Systems Biology Graphical Notation. *Nature Biotechnology*, 27, 735-741.
- Licata,L. Briganti,L. Peluso,D. Perfetto,L. Iannuccelli,M. Galeota,E. et al. (2012) MINT, the molecular interaction database: 2012 update, *Nucleic Acids Research*, 40, D857-D861.

Lin,B. Chen,Y. Chen X. Yu,Y. (2012) Comparison between JSON and XML in applications based on AJAX. In: International Conference On Computer Science and Service System, Nanjing, China, 11-13,1174-1177.

Nurseitov,N. Paulson,M. Reynolds,R. Izurieta,C. (2009) Comparison of JSON and XML Data Interchange Formats: A Case Study. CAINE 2009, 157-162.

Orchard,S. Ammari,M. Aranda,B. Breuza,L Briganti,L. Broackes-Carter,F. (2014) The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Research*, 42(1), D358-D36.

Perini,L. (2013) Diagrams In Biology. *The Knowledge Engineering Review*, 28, 273-286.

Schneck,C.M. (1996) Visual perception. *Occupational therapy for children*, 3rd edn. J. Case-Smith. Mosby, St. Louis, 357-385.

Seow,S. (2008) *Designing and Engineering Time: The Psychology of Time Perception in Software*. Addison-Wesley Professional, Boston, USA.

Sidiropoulos,K. Viteri,G. Sevilla,C. Jupe,S. Webber,M. Orlic-Milacic,M. et al. (2017) Reactome enhanced pathway visualization. *Bioinformatics*, 33(21), 3461–3467.

Wang,G. (2011) Improving data transmission in web applications via the translation between XML and JSON. In: *Third IEEE International Conference On Communications and Mobile Computing (CMC)*, Washington, DC, USA, 182-185.

6. THE GRAPH DATABASE

Reactome contains a detailed representation of cellular processes, as an ordered network of molecular reactions, interconnecting terms to form a graph of biological knowledge. Like many knowledge bases, Reactome uses a relational database to store its content. Although they are widely used for pathway data management, relational databases are not ideal for this purpose [Van Bruggen 2014; Vukotic et al. 2015]. Relational databases can be designed to represent and store with precision the many variables and annotation of complicated pathway information, but the final product will inevitably require intermediate tables to represent many-to-many relationships. As a result, database queries can require a high number of table join operations, which are both difficult to formulate and result in poor performance in terms of response times.

Reactome's data model comprises an interconnected entity/event network consisting of nodes connected by directed edges, that lends itself to be represented as a directed graph [Sedgewick and Wayne 2011]. Storing Reactome pathway data in this form has multiple benefits. Most significantly, it does not require transformation of the data into a flat or denormalised table format. Instead, it can be represented and maintained as originally represented in literature and annotated by Reactome curators, thereby reducing the complexity of the database and allowing a more straightforward and arguably more intuitive access to Reactome content [Vukotic et al. 2015].

This chapter summarises the content of Fabregat et al. 2018 (Publication P.4) which describes the motivation that led to the adoption of a graph database and details how Reactome benefits from this change in the underlying storage technology. The first section explains the underlying data model; the second

section describes how data was stored in a graph database and the quality assessment queries that were developed, having been made possible by the availability of the graph database and its associated query language. The third section shows the new software ecosystem built on top of the graph database. The graph database migration was conceived, designed and supervised by the author of the thesis; development work was shared in equal parts with Florian Korninger, a student intern who was later employed as a Reactome staff member.

6.1. REACTOME DATA MODEL

Reactome utilises a frame-based knowledge representation [Vastrik et al. 2007]. The data model (<https://reactome.org/content/schema>) consists of classes (frames) that describe different concepts such as reaction or entity. Classes have attributes (slots) that hold properties of the represented class instances, e.g. names and identifiers. The value types contained in the slots can be primitive (string, numbers, or boolean) or references to other class instances. Knowledge in Reactome is captured as instances of these classes with their associated attributes.

The Event and PhysicalEntity (PE) classes are core components of the Reactome data model. Events are the building blocks used in Reactome to represent biological processes and are further subclassed into Pathways and ReactionLikeEvents (RLE). RLEs are single-step molecular transformations. RLE includes Reaction among other types like FailedReaction, Polymerisation, Depolymerisation, and BlackBoxEvent. Although, all the examples discussed in this chapter involve transformations of the “Reaction” type, all types are handled in the same way with the same results. Ordered groups of RLEs, that together carry out a biological process, constitute

Pathways. PEs are the participants in these events. PE types include SimpleEntity for chemicals, EntityWithAccessionedSequence for proteins, Complex for multi-molecular structures and EntitySet for PEs grouped together on the basis of their shared function.

6.2. MOVING FROM A RELATIONAL TO A GRAPH DATABASE

Model persistence can be achieved with flat files, a relational database, or a non-relational database such as a graph database. The choice of storage mechanism determines how data are physically stored and accessed. Consequently, each of these options has strengths and weaknesses in terms of performance and scalability.

Until recently, Reactome relied on a MySQL relational database for both storing its content during curation and accessing it in the production phase. Factors that contributed to this decision included (i) use of ontology editor Protégé (<http://protege.stanford.edu>) as the curator tool during Reactome's early years with a Perl script processing the Protégé files to store content into a MySQL database, which was modelled according to the Protégé schema, (ii) at the time a relational database met Reactome's needs for data integrity and consistency, and (iii) relational databases were well established for biological data whereas graph based solutions were hardly used in the field [Have and Jensen 2013; Henkel et al. 2015].

As a result of these factors, when Reactome's relational database was implemented, the physical design was optimised for flexibility rather than performance. The relational design incorporated an increased level of abstraction to allow easier adoption of new concepts but at the expense of increased complexity and consequent query execution time. It should be underlined that as the graph database natively stores Reactome content in a

graph model, this trade-off between flexibility and performance is no longer necessary.

6.2.1. *THE GRAPH IMPORTER*

Graph databases have only recently become a popular technology for use in the area of computational biology. Henkel et al. proposed the concept of using graph databases for storage and retrieval of computational models representing biological systems [Henkel et al. 2015]. Summer et al. developed a Cytoscape application that takes advantage of the Neo4j database to perform server-side analysis of large and complex biological networks [Summer et al. 2015]. In [Lysenko et al. 2016] the authors explored the potential of using a graph database to facilitate data management and analysis to provide biological context to disease-related genes and proteins. Toure et al. developed a Java-based framework that transforms biological pathways represented in SBGN format into the Neo4j graph database, enabling more powerful management and querying of complex biological networks [Toure et al. 2016]. Balaur et al. demonstrated that advanced exploration of highly connected and comprehensive genome-scale metabolic reconstructions can benefit from an integrated graph representation of the model and associated data [Balaur et al. 2017]. Swainston et al. described *biochem4j*, which enables complex queries by linking a number of widely used chemical, biochemical and biology resources within a graph database [Swainston et al. 2017].

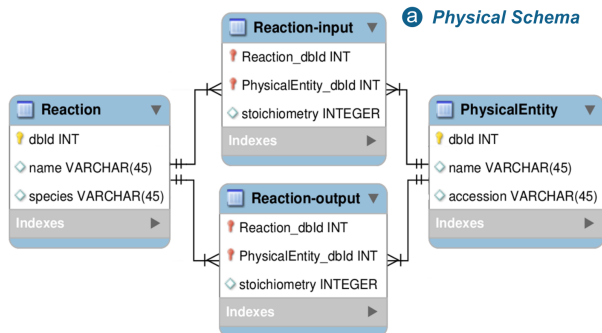
Recognising the potential value of graph databases, the author considered the possibility of gradually moving the production server from using the relational database to be based on a Neo4j graph database (<https://neo4j.com/>). Neo4j is an open source, transactional and ACID (Atomicity, Consistency, Isolation, and Durability) compliant graph database [Robinson et al. 2013]. Native graph databases, such as Neo4j, naturally store, manage, analyse, and use data

within the context of connections to improve performance and flexibility when handling highly interconnected data compared to that in SQL. Neo4j's greatest advantage and probably its most defining feature is Cypher: a declarative, pattern matching query language, specifically designed for dealing with graph data structures [Lal 2015; Neubauer et al. 2010].

Figure 6.1 provides a simplified example where reactions only contain lists of reactants and products, instances of the PE class. In the relational use case, two junction tables, Reaction-input and Reaction-output, are required to model these many-to-many relationships (Figure 6.1a). Each junction table contains foreign keys of the Reactions and the associated PEs. The SQL query to retrieve input and output entities of a given reaction requires two join operations per junction table (Figure 6.1b). In the first stage of its execution, each join operation forms the cartesian product between the tables and, during the filtering process, all rows of the result set that are not of interest are discarded.

The same structure of a reaction with inputs and outputs can be modelled in a simpler way with Neo4j, as exemplified by the reaction presented in Figure 6.1c. The reaction (green node), contains named outgoing relationships to corresponding input and output entities (purple nodes). Taking advantage of Cypher, the same query can be written in a shorter but more intuitive manner thanks to its ASCII-Art syntax [Vukotic et al. 2015] to represent patterns (Figure 6.1d). The query describes a pattern that includes a Reaction, identified by its identifier, with its associated outgoing input and output relationships. Finally, all nodes matching the specified pattern are returned.

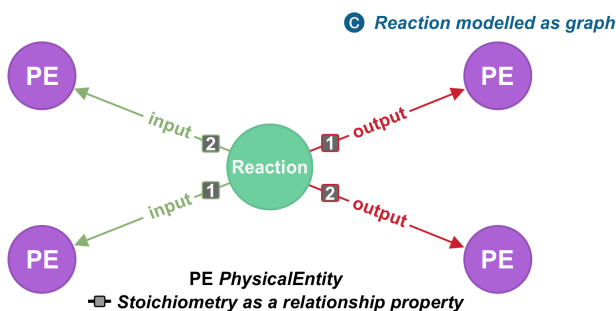
Relational Database



```
SELECT r.*, pe.* FROM Reaction r
JOIN Reaction-input ri ON r.dbId = ri.Reaction_dbId
JOIN PhysicalEntity pe ON pe.dbId = ri.PhysicalEntity_dbId
JOIN Reaction-output ro ON r.dbId = ro.Reaction_dbId
JOIN pe ON pe.dbId = ro.PhysicalEntity_dbId
WHERE r.dbId = IDENTIFIER
```

b SQL query

Graph Database



```
MATCH (r:Reaction{dbId:IDENTIFIER})-[:input|output]->(pe)
RETURN r,pe
```

d Cypher query

Figure 6.1. A simplified example where reactions only contain reactants and products represented by the class `PhysicalEntity`. (a) In the relational use case, two junction tables are required to model these many-to-many relationships (b) SQL query used to retrieve input and output entities of a given reaction where two join operations are needed per junction table. (c) The same reaction modelled as a graph. The reaction (green node) contains named outgoing relationships to corresponding input and output entities (purple nodes). (d) The same query written in Cypher illustrating a comparatively short and intuitive style [Fabregat et al. 2018].

The Reactome knowledgebase has many use cases similar to the example presented in Figure 6.1, where the use of a graph model combined with the Cypher query language can greatly improve response times and simplify the code required to access the data. For instance, recursively retrieving all the reactions of a pathway, retrieving the participants of a reaction or a pathway, deconstructing a complex or a set into its participating molecules, or enumerating the chain of consecutive reactions that lead to the formation of a signalling complex are common use cases that benefit from traversing the Reactome graph database rather than its MySQL counterpart.

Since their introduction in the 1970's, relational database engines have been optimised to provide efficient execution of SQL queries. This is particularly the case with global queries that aggregate large amounts of data without the need to perform any traversal operations. However, Reactome data contain many relationships, like those illustrated in Figure 6.1, and thus many join tables. Consequently, queries generally require traversal operations, a computationally intensive task that tends to result in poor performance compared to graph databases [Vicknair et al. 2010]. To address this issue and improve query performance, some resources have created redundant denormalised copies of their relational database [Birney et al. 2004, Eppig et al. 2015, Štefanič et al. 2015]. Nowadays, graph databases, such as Neo4j, offer a more appropriate alternative for cases of highly interconnected data.

The graph database batch importer (<https://github.com/reactome/graph-importer>) was developed to migrate the content from the relational database used in curation, to a graph database during each quarterly release process. Although the underlying data storage was changed, the original data model, also used in the MySQL database, was retained. The conversion was done following a depth-first approach starting from the top-level pathways and

traversing all the content, ensuring that each object is processed only once during the conversion. Every object constitutes a node in the graph and the edges that connect the nodes correspond to the names of the slots as defined in the domain model (Figure 6.2). The result is the generation of a Neo4j graph database that contains Reactome data. This can be directly queried by third parties using Cypher to retrieve target data.

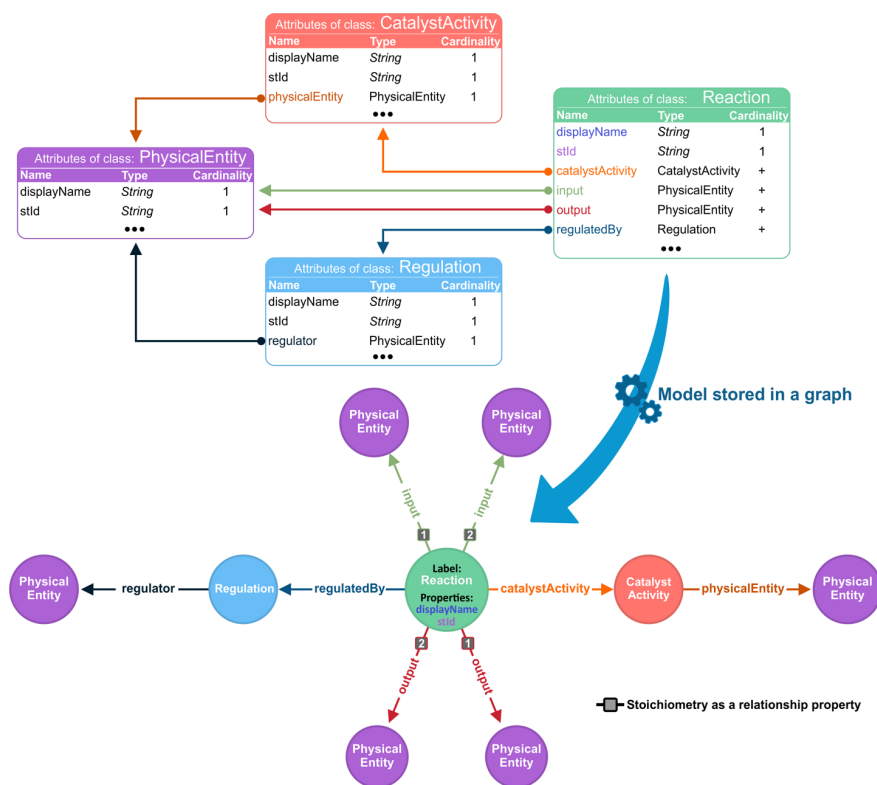


Figure 6.2. Representation of the content migration to the graph database. The example shows a Reaction class reduced to its inputs, outputs, catalyst and regulators. A model class instance is converted to a graph database node where (1) slots with primitive value types become node properties and (2) slots allocating instances of another class become relationships [Fabregat et al. 2018].

A number of tests were developed to ensure that both the Reactome graph and relational database have the same content after conversion. These tests are part of the graph-core and are executed after migration of the relational database to the graph database. Their purpose is to ensure that the data has been properly stored. The tests include checks to verify: (i) that the number of top level pathways present in the graph database corresponds to the number of those present in the relational database, (ii) that a given pathway in the graph database has the same ancestors as its counterpart in the relational database and (iii) that the content of a given complex is the same in both databases.

6.2.2. DATA INTEGRITY, CONSISTENCY AND QUALITY ASSESSMENT

Determining the quality of data provided by a company or a resource such as Reactome is challenging. Currently, most data quality measures are developed on an ad hoc basis to solve specific problems [Pipino et al. 2002] but in general there are three important aspects to consider: (i) data quality, (ii) data integrity and (iii) data consistency.

Data quality is a multi-dimensional concept. Data providers must consider both the subjective perceptions of individuals consuming the data and objective measurements based on the data set in question [Pipino et al. 2002]. Subjective data quality assessments reflect the needs and experiences of stakeholders (e.g. the collectors, custodians, and consumers of data products). Objective assessments can be task-independent or task-dependent. Task-independent metrics reflect states of the data without the contextual knowledge of the application, and can be applied to any data set, regardless of the tasks at hand. Task-dependent metrics, which take into account the organisation's business rules, company and government regulations, and constraints provided by the database administrator, are developed in specific

application contexts. Although these data quality metrics are out of the initial scope of this thesis, the development of the graph database provides the means to build a dashboard-like system where data quality could be measured following these principles.

Data consistency in database systems refers to the requirement that any given database transaction must change the relevant data only in allowed ways. Data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.

Data integrity refers to maintaining and ensuring the accuracy and consistency of data stored in a database, over its entire life-cycle. This is a critical aspect of the design, implementation and usage of any system which stores, processes, or retrieves data. In its broadest use, 'data integrity' refers to the accuracy and consistency of data stored in a database, data warehouse, data mart or other construct. The term 'data integrity' can be used to describe a state, a process or a function and is often used as a proxy for 'data quality'. Data with 'integrity' is said to have a complete or whole structure. Data values are standardised according to a data model and/or data type. All characteristics of the data must be correct (including business rules, relations, dates, definitions and lineage) for data to be complete. Data integrity is imposed within a database when it is designed and is authenticated through the ongoing use of error checking and validation routines. As a simple example, to maintain data integrity numeric columns/cells should not accept alphabetic data.

When data integrity and consistency concepts are correctly applied in a systematic way, the need of defensive programming when data is retrieved from the database is reduced or becomes unnecessary. Defensive programming

is a form of defensive design intended to ensure the continuing function of a piece of software under unforeseen circumstances. If data is systematically consistent across the database, when retrieving data for required slots does not require to check whether they are ‘null’ or not simply because that will not be the case. Avoiding defensive programming when consuming the database content reduces the code base keeping it easy to follow and therefore eases its future extension and maintenance.

```
MATCH (pre:ReactionLikeEvent)<-[:precedingEvent]-(rle:ReactionLikeEvent),
      (pre)-[:output|hasMember|hasCandidate*]->(o:PhysicalEntity),
      (rle)-[:input|hasMember|hasCandidate*]->(i:PhysicalEntity)
OPTIONAL MATCH (rle)-[:catalystActivity]->()-[:physicalEntity|hasMember|hasCandidate*]->(cp:PhysicalEntity)
OPTIONAL MATCH (rle)-[:regulatedBy]->()-[:regulator|hasMember|hasCandidate*]->(rp:PhysicalEntity)
WITH pre, rle, COLLECT(i) AS iss, COLLECT(cp) AS cps, COLLECT(rp) AS rps, collect(o) AS oss
WHERE NONE (x IN oss WHERE x IN iss)
      AND NONE(x IN oss WHERE x IN cps)
      AND NONE(x IN oss WHERE x IN rps)
OPTIONAL MATCH (a)-[:created]->(rle)
OPTIONAL MATCH (m)-[:modified]->(rle)
RETURN DISTINCT pre.stId AS PrecedingEvent, pre.displayName AS P_Name,
               rle.stId AS FollowingEvent, rle.displayName AS F_Name,
               a.displayName AS Created, m.displayName AS Modified
```

Figure 6.3. Example of data quality test written in Cypher. Reports Reaction class instances where none of its inputs, catalyst or regulators match with any of the outputs of its preceding event.

Once Reactome data was available in a Neo4j graph database, it became possible to use Cypher to query that data in an easier and more comprehensive way. In addition, the syntax of the new query language opened the door to systematically elaborate advanced queries to identify data inconsistencies and/or incomplete annotations. It is important to note that these queries could have been written against the relational database, but this was sufficiently challenging that it was never attempted. Construction of the appropriate SQL queries against the relational database built from Protégé files was a complicated task and Reactome developers found it necessary to write elaborate scripts to track down data quality issues. Appendix A contains a

subset of the Cypher queries developed for data quality checks over the graph database, namely:

- Event class instances which preceding events have an ongoing relationship pointing to them (Appendix A.1).
- Object pairs that reference each other, excluding 'author', 'created', 'edited', 'modified', 'revised', 'reviewed', 'inferredTo' and 'precedingEvent' slots, i.e. the objects have a cyclical relationship (Appendix A.2).
- CandidateSet class instances where 'hasMember' and 'hasCandidate' slots reference the same object (Appendix A.4). A given PhysicalEntity class instance cannot be member and candidate of a CandidateSet at the same time.
- CatalystActivity class instances where 'physicalEntity' and 'activeUnit' point to the same Complex class instance (Appendix A.5). The 'activeUnit' slot is not required in a CatalystActivity class instance unless it contains a subset of the object pointed in the 'physicalEntity' slot.
- Objects containing duplicated instances in multivalued slots that should not contain duplicates (i.e. members of an EntitySet or candidates in a CandidateSet) (Appendix A.5).
- Orphan events, i.e. events that have no hierarchical route connecting them to a top level pathway (Appendix A.6).
- ReactionLikeEvent class instances whose inputs, catalyst and regulators do not match any of the outputs of their annotated preceding events (Figure 6.3 and Appendix A.7). The 'precedingEvent' slot in a ReactionLikeEvent class instance is meant

to point to another instance of this class that occurs one step earlier in the series of events forming the pathway.

- Duplicated Complex and EntitySet class instances, i.e. two objects represent the same biological complex or set, in the same compartment (Appendix A.8). This happens when curators create an instance of a complex that already exists instead of reusing the existing one.
- Event class instances without regulator where the associated 'compartment' is not present in any of the participating molecules (Appendix A.9). The 'compartment' of a given reaction has to match with at least one of its participants.

The quality assessment project was first implemented for Reactome release 56 and has been run at each subsequent release. Reactome editorial staff use the results to report the identified instances to the curator responsible for checking and correction, to ensure high-quality content at each release.

6.2.3. NEW SOFTWARE ECOSYSTEM BASED ON THE GRAPH DATABASE

Before the graph database was made available, Reactome content was stored in MySQL and, thus, Reactome projects were built on top of this database. The GKInstance library for Java was used to access the data. This library was designed taking into account genericity over performance. It features a class called GKInstance that behaves as a placeholder for any class inheriting the abstract root class in the Reactome database (*DatabaseObject* - <https://reactome.org/content/schema/DatabaseObject>).

The GKInstance library allows developers to retrieve a set of objects belonging to a given class or sharing a set of properties. Additionally, single DatabaseObject class instances can be loaded using their database identifier (dbId). In both cases, these basic loading features only retrieved a handful of

single valued slots such as `dbId`, `displayName` or `schemaClass`. Once an object was loaded, retrieving the data in other fields required the developer to explicitly call a series of methods. To ensure success when calling these methods, the developer needed to write additional code to check whether the object `SchemaClass` was as expected and therefore would include the slots containing the data to be retrieved. In addition, most `GKInstance` methods handling data retrieval throw checked exceptions, making it mandatory to handle them. These constraints forced developers to write lengthy code for even the most basic actions over the MySQL database, creating an unnecessary boilerplate that hides the actual script purpose under a forced code scaffolding, resulting in very difficult to follow and maintain code.

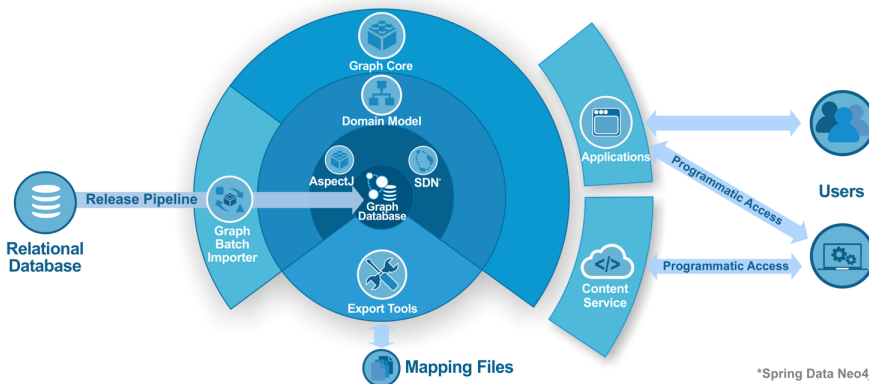


Figure 6.4. A schematic diagram of the new software ecosystem. The relational database is converted to a graph database via the batch importer that relies on the Domain Model. Spring Data Neo4j and AspectJ are two main pillars for the graph-core, which also rests on the Domain Model. Users access services or use tools that make direct use of the graph-core as a library that eliminates the code boilerplate for data retrieval and offers a data persistency mechanism. Finally, export tools take advantage of Cypher to generate flat mapping files [Fabregat et al. 2018].

After creating the graph database, the next requirement was to provide a new means of accessing the data, replacing the former GKInstance library. A new Java library named graph-core (<https://github.com/reactome/graph-core>) was developed to serve as a data access layer. The aim of this library was to provide easy access and data persistence as well as to reduce boilerplate code in third party projects that need to access and traverse Reactome content.

The graph-core uses Spring Data Neo4j (SDN) [Hunger 2016] to access graph content and AspectJ to enable lazy loading [Laddad 2010]. Lazy loading commonly refers to a design pattern that postpones the retrieval of object attributes until the point at which they are needed.

In this case, AspectJ weaver is used to intercept the getter methods and run specific code to silently retrieve more data when needed. Figure 6.4 presents a schematic illustration of the new Reactome graph database ecosystem.

The Content Service (<https://reactome.org/ContentService>) is a REST based web service [W3C 2004] built on top of the graph-core to provide programmatic access to the Graph Database for third party developers (<https://github.com/reactome/content-service>). Implemented using Spring MVC (<https://spring.io/>), the Content Service utilises the graph-core library and is fully documented with Open API (<https://www.openapis.org/>) (Figure 6.5). The Content Service provides equivalent methods to its predecessor and a set of new methods developed to cope with requests from external and internal users.

reactome

About Content Docs Tools Community Download

e.g. O95631, NTN1, signaling by EGFR, glucose **Go!**

Content Service ^{1.0}

RESTful service for Reactome content

Terms of service
 Reactome - Website
 Send email to Reactome
 Creative Commons Attribution 3.0 Unported License

- database** Reactome Data: Database info queries >
- discover** Reactome Data: Search engines discovery schema >
- diseases** Reactome Data: Disease related queries >
- entities** Reactome Data: PhysicalEntity queries >
- events** Reactome Data: Queries related to events >
- exporter** Reactome Data: Format Exporter >
- interactors** Molecule interactors >
- orthology** Reactome Data: Orthology related queries >
- participants** Reactome Data: Queries related to participants >
- pathways** Reactome Data: Pathway related queries >
- person** Reactome Data: Person queries >
- query** Reactome Data: Common data retrieval >

GET `/data/query/enhanced/{id}` More information on an entry in Reactome knowledgebase

POST `/data/query/ids` A list of entries in Reactome knowledgebase

POST `/data/query/ids/map` A list of entries with their mapping to the provided identifiers

GET `/data/query/{id}` An entry in Reactome knowledgebase

GET `/data/query/{id}/{attributeName}` A single property of an entry in Reactome knowledgebase

- references** Reactome xRefs: ReferenceEntity queries >
- schema** Reactome Data: Schema class queries >
- search** Reactome Search >
- species** Reactome Data: Species related queries >

Models >

Figure 6.5. Content Service documentation page. The methods available are grouped under categories such as entities, events, orthology, participants, query, etc. This view shows the methods available under the query category. Each method is fully documented. When expanded, OpenAPI offers the possibility of using them directly in the interface, for testing purposes (<https://reactome.org/ContentService/#!/query>).

6.2.4. RESULTS AND DISCUSSION

One of the main advantages of this new solution is that it is faster and less computationally intensive than previous methods that were based on the relational database. Performing queries against the graph database constitutes a more scalable approach, resulting in higher throughput and ultimately a more robust Content Service, able to cope with an ever-increasing number of requests. Additionally, the resulting product is easier to maintain; most new methods can be added by simply writing the respective Cypher queries, avoiding the need to write complex algorithms (Figure 6.1b).

a. Which molecules participate in Interleukin-4 and 13 signaling (R-HSA-6785807)?

```
MATCH (p:Pathway{stId:"R-HSA-6785807"})-[:hasEvent*]->(rle:ReactionLikeEvent),
      (rle)-[:input|output|catalystActivity|entityFunctionalStatus|physicalEntity|regulatedBy|
            regulator|hasComponent|hasMember|hasCandidate|repeatedUnit*]->(pe:PhysicalEntity),
      (pe)-[:referenceEntity]->(re:ReferenceEntity)-[:referenceDatabase]->(rd:ReferenceDatabase)
RETURN DISTINCT re.identifier AS Identifier, rd.displayName AS Database
```

b. In which pathways does CCR5 (UniProt:P51681) participate?

```
MATCH (p:Pathway)-[:hasEvent*]->(rle:ReactionLikeEvent),
      (rle)-[:input|output|catalystActivity|entityFunctionalStatus|physicalEntity|regulatedBy|
            regulator|hasComponent|hasMember|hasCandidate|repeatedUnit*]->(pe:PhysicalEntity),
      (pe)-[:referenceEntity]->(re:ReferenceEntity{identifier:"P51681"}),
      (re)-[:referenceDatabase]->(rd:ReferenceDatabase{displayName:"UniProt"})
RETURN DISTINCT p.stId AS Identifier, p.displayName AS Pathway
```

Figure 6.6. Examples of frequent use cases that can be answered using Cypher queries. (a) Retrieving the participating molecules for “Interleukin-4 and 13 signalling” pathway. (b) Retrieving the pathways in which CCR5 participates [Fabregat et al. 2018].

The use cases in Figure 6.6 aim to emphasise how Reactome data queries have been simplified by the adoption of the graph database and they are available as methods in the Content Service API (<https://reactome.org/ContentService/>). In particular, the query in Figure 6.6a shows how participating molecules for a pathway can be retrieved. The reverse query, identifying pathways where a molecule participates, is shown in Figure 6.6b. It uses a similar pattern to Figure 6.6a, but fixes the molecule

identifier for the reference entity so the pathways containing it can be populated after traversing.

To assess the improvements, a set of stress tests was designed to measure the impact of adopting the graph database in Reactome. All stress tests were executed on a standard laptop featuring an Intel Core i7 at 2.6 GHz, 16 GB of DDR3 memory at 1,600 MHz, and 256 GB of flash storage. The tests do not aim to compare the two storage technologies (MySQL and Neo4j) but instead their usage by Reactome. The stress tests were run against the web services build on top of each storage technology and included two scenarios: (i) simulation of one user sequentially querying 5,000 reactions for Homo sapiens and (ii) simulating an increasing set of users simultaneously performing the previous task. In each case the resulting data for every reaction had to be marshalled as an instance of the correspondent model class. The test comprised four executions; two against the previous web service running on top of the relational database and two accessing the new web service running on top of the graph database through the newly created graph-core library (<https://github.com/reactome/graph-core>). Reactions were accessed in a sequential fashion to ensure that caching would not provide an advantage for either approach. Sequential access ensures that a queried object will never be retrieved more than once in the same test. It should be mentioned that prior to execution of the stress tests both Neo4j and MySQL databases were configured to allocate 50% of the available physical memory (8 GB).

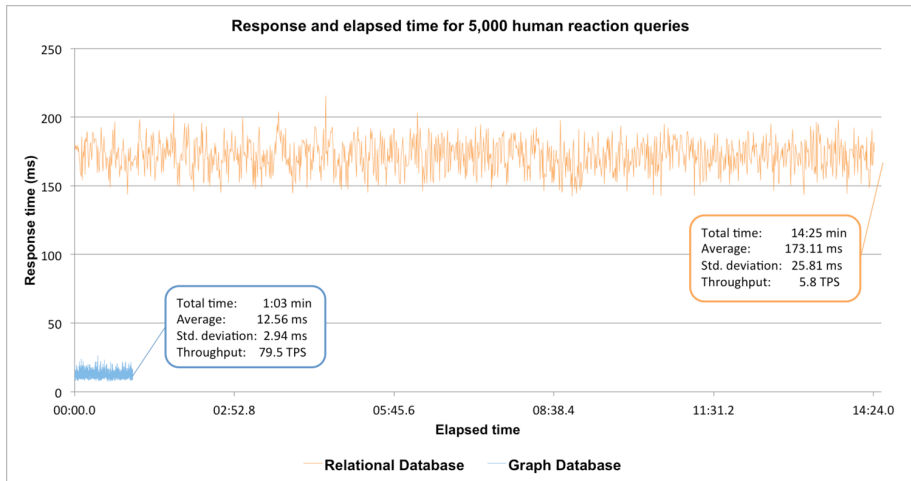


Figure 6.7. Comparison of the response and elapsed time for one user sequentially retrieving 5,000 reaction instances from the graph and relational databases (blue and orange respectively). The graph database software ecosystem achieved a 93% average improvement in performance compared to that of the relational database [Fabregat et al. 2018].

As illustrated in Figure 6.7, querying the data stored in the relational database resulted in significantly longer response times. In particular, in the case of the relational implementation of the Reactome knowledgebase, the average query time was 173.11 ms (± 25.81) while in the case of the graph implementation, the average response time dropped to 12.56 ms (± 2.94), a 93% reduction in average query time. The new implementation supported higher throughput in terms of transactions per second (TPS), reaching 79.5 TPS compared to 5.8 TPS. As a result of this boost in performance, all 5,000 queries to the graph database were performed in 63 seconds; the relational implementation required more than 14 minutes to complete the same task.

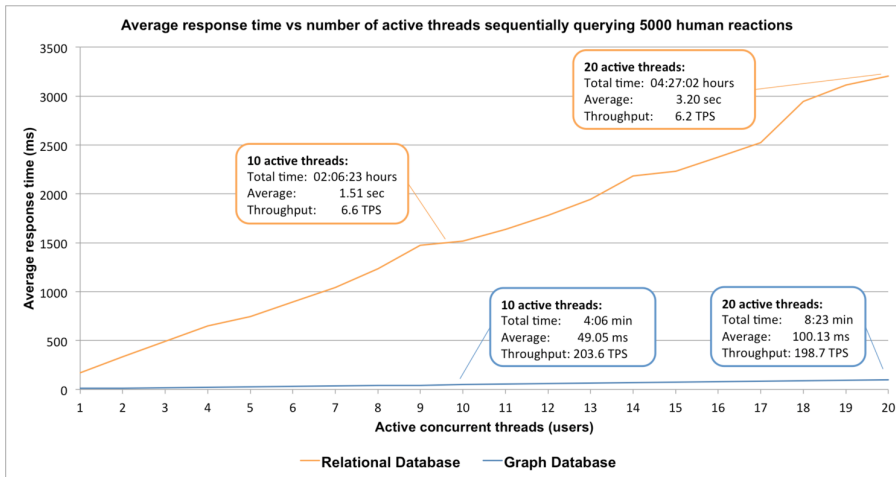


Figure 6.8. Response time versus an increasing set of users simultaneously performing queries for 5,000 reaction instances. Starting with one and scaling up to 20 concurrent users, the relational database performance drops while the graph database keeps a low response time and a good throughput as the number of active threads increases [Fabregat et al. 2018].

A second stress test simulated a more realistic scenario where multiple users perform concurrent database queries (Figure 6.8). Once again, querying the Reactome knowledgebase in its relational implementation resulted in significantly longer response times. For instance, in case of 10 concurrent threads performing queries to the relational implementation of the Reactome knowledgebase the average response time was 1,516 ms, while in the case of the graph implementation, the average response time dropped to 49.05 ms. In addition, the new implementation achieved higher throughput reaching 203.6 TPS compared to 6.6 TPS. Consequently, the graph implementation of Reactome provides higher scalability enabling Reactome to handle larger volumes of user requests.

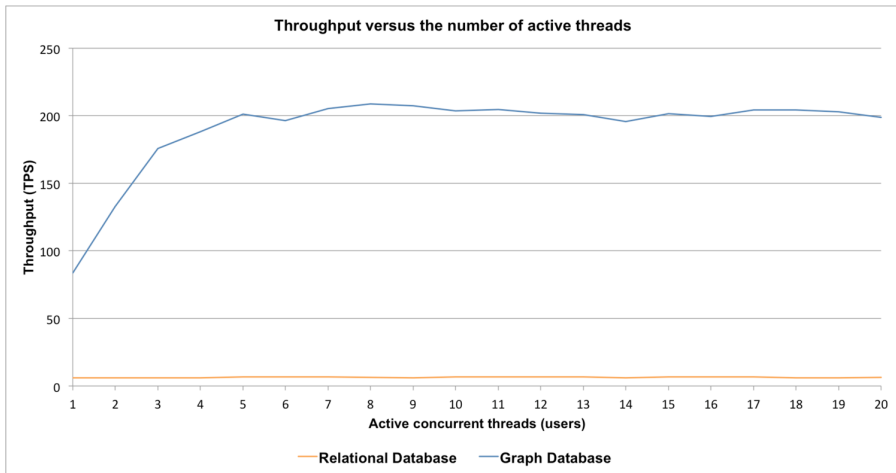


Figure 6.9. Throughput measured in transactions per second, versus the number of users concurrently performing queries for 5,000 reaction instances in Homo sapiens [Fabregat et al. 2018].

Figure 6.9 presents a comparison between the throughputs achieved by both systems against the number of users performing concurrent queries. The graph implementation achieved a higher number of transactions per second that reached a plateau after the point where the number of active threads becomes equal to the available processor cores, in this case 4. In comparison, the measured throughput of the relational implementation does not seem to be improved by the availability of multiple processor cores.

Reactome usage statistics show that a growing number of users download the Reactome graph database. Based on questions received by Reactome's help desk service, it is believed that users are regularly performing local Cypher queries against the complete Reactome knowledgebase. During the first year that the Reactome graph database was available there were 2,385 downloads by 912 unique users. 118 of these users downloaded the graph database after each data release. It is noteworthy that at the time of writing the size of the

Reactome relational database (release v63) is around 2.1 GB while the size of the graph database is approximately 1.9 GB. Figure 6.10 provides a summary of the graph database.

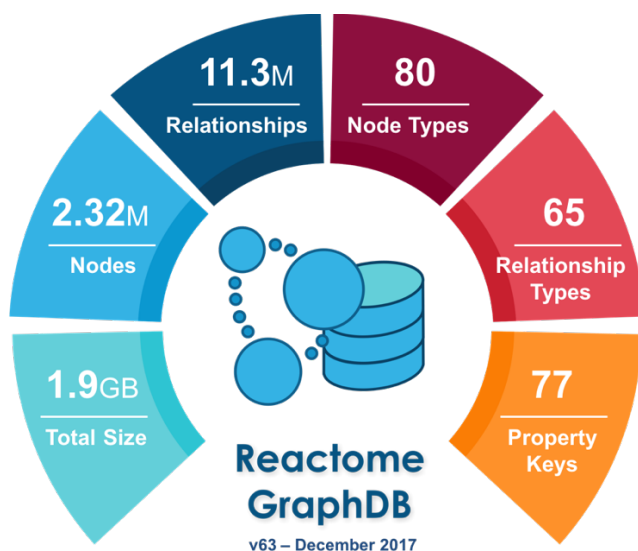


Figure 6.10. The Reactome graph database in numbers for the total size, the number of nodes, relationships, node types, relationship types and property keys (adapted from Fabregat et al. 2018, updating the numbers for version 63).

With a powerful tool for managing highly connected data sets and complex queries at its disposal, Reactome is able to provide faster and more stable services to researchers around the world. In the near future, Reactome plans to upgrade its services and leverage the full potential of Cypher to answer questions that require deeper examination of its data. In particular, the integration of a graph database lowers the complexity of problems that require traversing the knowledgebase, such as identifying causal interactions or revealing all possible paths between two molecules.

The creation of the graph database has also made possible new set of data quality checks that have revealed previously-unidentified data consistency issues. Once visible, Reactome curators have been able to address the issues and eliminate them before making the data public.

Future development in Reactome is unlikely to be negatively influenced by the fact that Neo4j is by nature schema-less, mainly because the rigid schema of Reactome relational database, with all the applied constraints, is used to ensure data consistency during the curation phase. As already mentioned, currently, data are migrated to Neo4j during each quarterly release process and are used to speed up queries in production.

In conclusion, through the adoption of the Neo4j graph database, which allows the power of its query language to be harnessed, Reactome provides efficient access to its pathway knowledgebase. As a result of this shift in the underlying data storage technology, the average query time has been reduced by up to 93%. In addition, the graph-core library and the Content Service leverage the benefits of this shift and can be used by third party applications to efficiently access Reactome.

6.3. SUMMARY

The Reactome data model perfectly fits with the definition of a directed graph. By migrating its content to a Neo4j graph database, the resource could benefit of the advantages that this technology offers to better support genome analysis, modelling and systems biology. Therefore, the graph-importer was developed to migrate the Reactome content from the relational database used in curation to a graph database during each quarterly release process. This conversion was done following a depth-first approach starting from the top-

level pathways and traversing all the content, ensuring that each object is processed only once during the conversion.

A new software ecosystem was developed on top of the graph database. A new java library, graph-core, was developed to act as the data access layer and cope with the concept of data persistence. This library transparently retrieves data to make it available when needed and reduces the boilerplate code. A new RESTful service, named Content Service, was developed to provide programmatic access to the community, containing similar methods to its predecessor plus a new set of advanced methods to cope with users requests.

The new graph database has two main advantages; higher performance and simpler ways to perform complex queries. This has also made possible new set of data quality checks that have revealed previously-unidentified data consistency issues. Reactome now provides efficient access to its pathway knowledgebase and as a result of this shift in the underlying data storage technology, the average query time has been reduced up to 93%. In addition, the graph-core library and the Content Service leverage these benefits of this shift and can be used by third party applications to efficiently access Reactome.

6.4. REFERENCES

- Balaur,I. Mazein,A. Saqi,M. Lysenko,A. Rawlings,C.J. Auffray,C. (2016) Recon2Neo4j: applying graph database technologies for managing comprehensive genome-scale networks. *Bioinformatics*, 33(7), 1096-1098.
- Birney,E. Andrews,D. Bevan,P. Caccamo,M. Cameron,G. Chen,Y. et al. (2004) Ensembl 2004. *Nucleic Acids Research*, 32, D468-D470.

- Eppig,J.T. Blake,J.A. Bult,C.J. Kadin,J.A. Richardson,J.E. (2015) Mouse Genome Database Group. The Mouse Genome Database (MGD): facilitating mouse as a model for human biology and disease. *Nucleic Acids Research*, 43, D726-D736.
- Fabregat,A. Korninger,F. Viteri,G. Sidiropoulos,K. Marin-Garcia,P. Ping,P. et al. (2018) Reactome Graph Database: Efficient access to complex pathway data. *PLOS Computational Biology*, 14(1), 1-13.
- Fabregat,A. Sidiropoulos,K. Garapati,P. Gillespie,M. Hausmann,K. Haw,R. et al. (2015) The Reactome pathway Knowledgebase. *Nucleic Acids Research*, 44, D481-D487.
- Have,C.T. and Jensen,L.J. (2013) Are graph databases ready for bioinformatics? *Bioinformatics*, 29(24), 3107.
- Henkel,R. Wolkenhauer,O. Waltemath,D. (2015) Combining computational models, semantic annotations and simulation experiments in a graph database. *Database (Oxford)*, 2015, 1-8.
- Hunger,M. (2012) *Good Relationships: The Spring Data Neo4j Guide Book*. InfoQ enterprise software development. C4Media.
- Laddad,R. (2010) *AspectJ in Action*. 2nd edn. Manning Publications Co., Greenwich, Connecticut, USA.
- Lal,M. (2015) *Neo4j graph data modeling*. Packt Publishing Ltd., Birmingham, UK.
- Lysenko,A. Roznovat,I.A. Saqi,M. Mazein,A. Rawlings,C.J. Auffray,C. (2016) Representing and querying disease networks using graph databases. *BioData Mining*, 9(1), 23.

Neubauer,P. (2010) Graph Databases, NOSQL and Neo4j. InfoQ. 12 May 2010. Available from: <https://www.infoq.com/articles/graph-nosql-neo4j>. Cited 15 January 2017.

Pipino,L.L. Lee,Y.W. Wang,R.Y. (2002) Data quality assessment. Communications of the ACM, 45(4), 211-218.

Robinson,I. Webber,J. Eifrem,E. (2013) Graph Databases. O'Reilly Media Inc., Sebastopol, California, USA.

Sedgewick,R. and Wayne,K. (2011) Algorithms. 4th edn. Addison-Wesley, Professional, Boston, USA, 566-596.

Štefanič,S. Lexa,M. (2015) A Flexible Denormalization Technique for Data Analysis above a Deeply-Structured Relational Database: Biomedical Applications. In: Ortuño F, Rojas I, editor. Bioinformatics and Biomedical Engineering. IWBBIO 2015. Lecture Notes in Computer Science. Springer, 9043.

Summer,G, Kelder,T. Ono,K. Radonjic,M. Heymans,S. Demchak,B. (2015) cyNeo4j: connecting Neo4j and Cytoscape. Bioinformatics, 31(23), 3868-3869.

Swainston,N. Batista-Navarro,R. Carbonell,P. Dobson,P.D. Dunstan,M. Jervis,A.J. et al. (2017) biochem4j: Integrated and extensible biochemical knowledge through graph databases. PLoS ONE, 12(7), e0179130.

Touré,V. Mazein,A. Waltemath,D. Balaur,I. Saqi,M. Henkel,R. et al. (2016) STON: exploring biological pathways using the SBGN standard and graph databases. BMC Bioinformatics, 17, 494.

- Van Bruggen,R. (2014) Learning Neo4j. Packt Publishing Ltd., Birmingham, UK.
- Vastrik,I. D'Eustachio,P. Schmidt,E. Joshi-Tope,G. Gopinath,G. Croft,D. et al. (2007) Reactome: a knowledge base of biologic pathways and processes. *Genome Biology*, 8, R39.
- Vicknair,C. Macias,M. Zhao,Z. Nan,X. Chen,Y. Wilkins,D. (2010) A comparison of a graph database and a relational database: a data provenance perspective. *Proceedings of the 48th Annual Southeast Regional Conference*, New York, USA.
- Vukotic,A. Watt,N. Abedrabbo,T. Fox,D. Partner,J. (2014) Neo4j in Action. 1st edn. Manning Publications Co., Greenwich, Connecticut, USA. 2014.
- World Wide Web Consortium (W3C) Web Services Architecture, W3C Working Group Note. 11 February 2004. Available from: <https://www.w3.org/TR/ws-arch/>. Cited 20 June 2017.

7. CONCLUSIONS

The work presented in this thesis approaches the performance optimisation of biological pathway data storage and retrieval as well as its interactive visualisation. Different problems for pathway data storage, analysis and visualisation have been addressed by studying and applying a variety of techniques and concepts in both server and client side. Although the proposed strategies, development frameworks, data structures, libraries and storage management system have been applied to Reactome, they can also be applied to other similar resources in order to achieve analogous results in performance and visualisation improvement.

The Pathway Browser is Reactome's primary means of viewing and interacting with its data. A new version of it was re-engineered to provide a responsive, easy to use, long-lasting and large-scale tool. The main goal was offering a good user experience by reducing the loading time and providing a more attractive user interface. The new version of the Pathway Browser was developed using the MVP software layout pattern. Excessive memory usage was avoided by loading objects on demand and caching them in an LRU list with a maximum capacity defined. A State Manager was put in place to maintain the internal application state and allow users to take advantage of the existing browser history management features. Third party widgets were used when available, and a number of modules were developed as widgets. These retrieve data from third party resources and show their content based on items selected in the Pathway Browser.

Through the use of highly optimised, in-memory data structures and algorithms, a stable, high performance pathway analysis service was developed to enable the analysis of genome-wide datasets within seconds,

allowing interactive exploration and analysis of high throughput data. This was achieved by splitting the pathway analysis method in four steps, in a way that every challenge can be easily addressed in polynomial time using the appropriate data structures, speeding up the process and minimising the memory usage, so the whole data structure can be kept in memory for a high-performance analysis. The result is a new set of Analysis Tools which vastly improve Reactome analysis interface performance and stability. A new RESTful web service was developed to support the high-throughput pathway analysis. This service is documented in detail and allows the use of the Reactome server for batch dataset analysis. A pathway analysis data submission interface was integrated into the Pathway Browser to facilitate users performing analysis and display the results in that tool.

A representation of the complex parent-child relationships present in Reactome's hierarchical organisation was created to provide a means of easily navigating these data and overlaying analysis result in a way that the user can easily distinguish the most significant areas of biology represented in their data. A custom radial layout algorithm was developed to produce the desired result. Finally, a widget was developed to interactively render the Pathways Overview and overlay the analysis results.

A new version of the pathway Diagram Viewer was developed to boost performance and improve usability through the use of highly optimised data structures and algorithms. The new version provides a robust, scalable solution to pathway visualisation that is easily integrated into third party applications. It accomplishes loading and rendering of 97% of the diagrams in Reactome in less than 1 s (versus 57% previously); 74% of the total number of diagrams are loaded and rendered in under 0.5 s (versus 31% previously). Visual appeal and navigability issues of previous Reactome diagrams have

been addressed by the introduction of Enhanced High Level Diagrams, which represent biological processes in a familiar textbook style that allows intuitive navigation to more specific subtopics. In addition, the Reactome pathway iconography library provides graphical representations of common molecular biology elements suitable for use in slides and publications. Finally, classic pathway diagrams can now be exported in PPTX format, allowing their editing and reuse with familiar presentation software.

The Reactome content was moved from a relational database to a graph database (Neo4j) to benefit from the advantages that this technology offers to better support genome analysis, modeling and systems biology. A new software ecosystem was developed on top of the new graph database where the graph-core, a java library, is the data access layer that copes with the concept of data persistence. A new RESTful service, the Content Service, was also made available to provide programmatic access to the community. These changes result in an efficient access to Reactome content and as a result of this shift in the underlying data storage technology, the average query time has been reduced up to 93%. In addition, the graph-core library and the Content Service leverage the benefits of this shift and can be used by third party applications to efficiently access Reactome.

As a consequence of this work there is a plan to modify the pathways analysis tools to allow the use of post-translational modifications to differentiate different variants of same protein. The Pathways Overview and Diagram Viewer widgets will also be updated to reflect these changes. The graph-core will focus on updating the SDN version and integrating interaction data from IntAct (<http://www.ebi.ac.uk/intact/>).

Finally, the achievements and conclusions in this thesis demonstrate the major impact achieved with the reengineering of the main components of Reactome,

including moving the Reactome public database to a graph database and the software layer to manage these data. Although this resulted in a great boost in performance and improved visualisation, this should be considered as an initial milestone, setting a robust foundations for further development and the potential for entirely new functionalities in the growing 'omics ecosystem, pathways databases and network-oriented biomolecular data resources. The author would like to invite the community to use the open data Reactome graph database and the associated infrastructure improvements to develop their own novel uses of Reactome data and consider the software engineering approaches discussed above.

A. QUERIES FOR DATA QUALITY ASSESSMENT

Tests for data integrity are performed when the content of the relational database is converted to a graph database using the graph-importer. Once the graph database is available, the graph-qa tests for data consistency and quality tests are executed. As the code might be included in a future publication, at present the graph-qa project is a private project in the Reactome GitHub repository (<https://github.com/reactome/graph-qa>). To counteract the accessibility problem, this appendix contains a subset of Cypher queries developed for data quality checks over the graph database.

```
MATCH (n:Event)-[r:precedingEvent]->(x:Event)-->(n)
OPTIONAL MATCH (a)-[:created]->(n)
OPTIONAL MATCH (m)-[:modified]->(n)
RETURN DISTINCT n.dbId AS DbIdA, n.stId AS StIdA, n.displayName AS NameA,
                x.dbId AS DbIdB, x.stId AS StIdB, x.displayName AS NameB,
                a.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, StIdA, DbIdA, StIdB, DbIdB
```

A.1. Event class instances which preceding events have an ongoing relationship pointing to them.

```
MATCH (n)-->(x)-->(n)
WHERE NOT (n)-[:author|created|edited|modified|revised|reviewed|inferredTo|precedingEvent]-(x)
OPTIONAL MATCH (a)-[:created]->(n)
OPTIONAL MATCH (m)-[:modified]->(n)
RETURN DISTINCT n.dbId AS DbIdA, n.stId AS StIdA, n.displayName AS NameA,
                x.dbId AS DbIdB, x.stId AS StIdB, x.displayName AS NameB,
                a.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, StIdA, DbIdA, StIdB, DbIdB
```

A.2. Object pairs that reference each other, excluding 'author', 'created', 'edited', 'modified', 'revised', 'reviewed', 'inferredTo' and 'precedingEvent' slots, i.e. the objects have a cyclical relationship.

```

MATCH (es:EntitySet)-[:hasMember]->(pe:PhysicalEntity)<-[:hasCandidate]->(es)
OPTIONAL MATCH (a)-[:created]->(es)
OPTIONAL MATCH (m)-[:modified]->(es)
RETURN DISTINCT es.dbId AS DbIdA, es.stId AS StIdA, es.displayName AS NameA,
                pe.dbId AS DbIdB, pe.stId AS StIdB, pe.displayName AS NameB,
                a.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, StIdA, DbIdA, StIdB, DbIdB

```

A.3. CandidateSet class instances where 'hasMember' and 'hasCandidate' slots reference the same object (a given PhysicalEntity class instance cannot be member and candidate of a CandidateSet).

```

MATCH (ca:CatalystActivity)-[:physicalEntity]->(c:Complex)<-[:activeUnit]->(ca)
OPTIONAL MATCH (a)-[:created]->(ca)
OPTIONAL MATCH (m)-[:modified]->(ca)
RETURN DISTINCT ca.dbId AS CatalystID, ca.displayName AS CatalystActivity,
                c.stId AS ComplexID, c.displayName AS Complex,
                a.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, CatalystID, ComplexID

```

A.4. CatalystActivity class instances where 'physicalEntity' and 'activeUnit' point to the same Complex class instance ('activeUnit' is not required unless it is a subset of the 'physicalEntity').

```

MATCH (a:DatabaseObject)-[r]->(b)
WHERE r.stoichiometry > 1 AND
      NOT TYPE(r) IN ["input", "output", "hasComponent", "hasModifiedResidue"]
OPTIONAL MATCH (c)-[:created]->(a)
OPTIONAL MATCH (m)-[:modified]->(a)
RETURN DISTINCT a.dbId AS DbIdA, a.stId AS StIdA, a.displayName AS NameA,
                TYPE(r) AS Relationship,
                b.dbId AS DbIdB, b.stId AS StIdB, b.displayName AS NameB,
                c.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, StIdA, DbIdA, StIdB, DbIdB

```

A.5. Objects containing duplicated instances in multivalued slots that should not contain duplicates (i.e. members of an EntitySet or candidates in a CandidateSet).

```

MATCH (e:Event{isInferred:False})
WHERE NOT (e:TopLevelPathway) AND
      NOT (:TopLevelPathway)-[:hasEvent*]->(e) AND
      NOT (e)-[:inferredTo]->(:Event)
OPTIONAL MATCH (a)-[:created]->(e)
OPTIONAL MATCH (m)-[:modified]->(e)
RETURN DISTINCT e.dbId AS DbId, e.displayName AS Name, e.stId as StId,
      a.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, StId, DbId

```

A.6. Orphan events, i.e. events that have no hierarchical route connecting them to a top level pathway.

```

MATCH (pre:ReactionLikeEvent)<-[:precedingEvent]-(rle:ReactionLikeEvent),
      (pre)-[:output|hasMember|hasCandidate*]->(o:PhysicalEntity),
      (rle)-[:input|hasMember|hasCandidate*]->(i:PhysicalEntity)
OPTIONAL MATCH (rle)-[:catalystActivity]->()-[:physicalEntity|hasMember|hasCandidate*]->(cp:PhysicalEntity)
OPTIONAL MATCH (rle)-[:regulatedBy]->()-[:regulator|hasMember|hasCandidate*]->(rp:PhysicalEntity)
WITH pre, rle, COLLECT(i) AS iss, COLLECT(cp) AS cps, COLLECT(rp) AS rps, collect(o) AS oss
WHERE NONE (x IN oss WHERE x IN iss)
      AND NONE(x IN oss WHERE x IN cps)
      AND NONE(x IN oss WHERE x IN rps)
OPTIONAL MATCH (a)-[:created]->(rle)
OPTIONAL MATCH (m)-[:modified]->(rle)
RETURN DISTINCT pre.stId AS PrecedingEvent, pre.displayName AS P_Name,
      rle.stId AS FollowingEvent, rle.displayName AS F_Name,
      a.displayName AS Created, m.displayName AS Modified

```

A.7. Reaction class instances whose inputs, catalyst and regulators do not match any of the outputs of their preceding events.

```

MATCH (n1c)-[:compartment]-(n1:Complex)-[:hasComponent]->(pe:PhysicalEntity),
      (pe)-[:hasComponent]-(n2:Complex)-[:compartment]->(n2c)
WHERE NOT ()-[:inferredTo]->(n1) AND NOT ()-[:inferredTo]->(n2) AND NOT n1 = n2
WITH DISTINCT n1, n2
MATCH (n1)-[r1:hasComponent]->(n1pe:PhysicalEntity),
      (n2)-[r2:hasComponent]->(n2pe:PhysicalEntity)
WITH n1, COLLECT(DISTINCT {pe: n1pe, n: r1.stoichiometry}) AS n1pes,
      n2, COLLECT(DISTINCT {pe: n2pe, n: r2.stoichiometry}) AS n2pes
WHERE ALL(c IN n1pes WHERE c IN n2pes) AND ALL(c IN n2pes WHERE c IN n1pes)
OPTIONAL MATCH (an1)-[:created]->(n1)
OPTIONAL MATCH (an2)-[:created]->(n2)
RETURN DISTINCT n1.stId AS C1, n1.displayName AS C1Name, an1.displayName AS C1Created,
               n2.stId AS C2, n2.displayName AS C2Name, an2.displayName AS C2Created
ORDER BY C1Created, C1

```

A.8. Duplicated Complex class instances, i.e. two objects represent the same biological complex in the same compartment.

```

MATCH (rle:ReactionLikeEvent)
WHERE NOT (rle:BlackBoxEvent) AND NOT (rle)-[:regulatedBy]->() AND (rle)-[:compartment]->()
WITH DISTINCT rle
MATCH (rlec:Compartment)-[:compartment]-(rle),
      (rle)-[:input|output|requiredInputComponent|catalystActivity|entityFunctionalStatus
            |physicalEntity|hasComponent|hasMember|hasCandidate|repeatedUnit*]->(pe:PhysicalEntity),
      (pe)-[:compartment]->(pec:Compartment)
WITH rle, COLLECT(DISTINCT rlec.displayName) AS rlecs, COLLECT(DISTINCT pec.displayName) AS peccs
WHERE ANY(c IN rlecs WHERE NOT c IN peccs)
OPTIONAL MATCH (a)-[:created]->(rle)
OPTIONAL MATCH (m)-[:modified]->(rle)
RETURN DISTINCT rle.stId AS Identifier, rle.displayName AS Name,
               a.displayName AS Created, m.displayName AS Modified
ORDER BY Created, Modified, Identifier

```

A.9. Event class instances without regulator where the associated 'compartment' is not present in any of the participating molecules

P. PUBLICATIONS DIRECTLY RELATED TO THE PRESENTED WORK

P.1. REACTOME PATHWAY ANALYSIS: A HIGH-PERFORMANCE IN-MEMORY APPROACH

Fabregat et al. *BMC Bioinformatics* (2017) 18:142
DOI 10.1186/s12859-017-1559-2

BMC Bioinformatics

SOFTWARE

Open Access

Reactome pathway analysis: a high-performance in-memory approach



Antonio Fabregat^{1,2}, Konstantinos Sidiropoulos¹, Guilherme Viteri¹, Oscar Forner¹, Pablo Marin-Garcia^{3,4}, Vicente Arnaiz^{5,6}, Peter D'Eustachio⁷, Lincoln Stein^{8,9} and Henning Hermjakob^{1,10*}

Abstract

Background: Reactome aims to provide bioinformatics tools for visualisation, interpretation and analysis of pathway knowledge to support basic research, genome analysis, modelling, systems biology and education. Pathway analysis methods have a broad range of applications in physiological and biomedical research; one of the main problems, from the analysis methods performance point of view, is the constantly increasing size of the data samples.

Results: Here, we present a new high-performance in-memory implementation of the well-established over-representation analysis method. To achieve the target, the over-representation analysis method is divided in four different steps and, for each of them, specific data structures are used to improve performance and minimise the memory footprint. The first step, finding out whether an identifier in the user's sample corresponds to an entity in Reactome, is addressed using a radix tree as a lookup table. The second step, modelling the proteins, chemicals, their orthologous in other species and their composition in complexes and sets, is addressed with a graph. The third and fourth steps, that aggregate the results and calculate the statistics, are solved with a double-linked tree.

Conclusion: Through the use of highly optimised, in-memory data structures and algorithms, Reactome has achieved a stable, high performance pathway analysis service, enabling the analysis of genome-wide datasets within seconds, allowing interactive exploration and analysis of high throughput data. The proposed pathway analysis approach is available in the Reactome production web site either via the AnalysisService for programmatic access or the user submission interface integrated into the PathwayBrowser. Reactome is an open data and open source project and all of its source code, including the one described here, is available in the AnalysisTools repository in the Reactome GitHub (<https://github.com/reactome/>).

Keywords: Pathway analysis, Over-representation analysis, Data structures

Background

Reactome (<http://reactome.org>) is a free, open-source, curated and peer-reviewed knowledge-base of biomolecular pathways. It aims to provide bioinformatics tools for visualisation, interpretation and analysis of pathway knowledge to support basic research, genome analysis, modelling, systems biology and education.

Nowadays, pathway analysis methods have a broad range of applications in physiological and biomedical

research. On the one hand, based on a given dataset, these methods help researchers to discover which areas of biology, and biomolecules, are crucial to understand the phenomena under study. On the other hand, pathway analysis methods should never be taken as black boxes from where experimental data goes in, and true statements come out, but perhaps more as metal detectors in haystacks helping researchers to find biologically meaningful needles [1].

Pathway analysis methods are mainly used to analyse Omics data obtained from high-throughput technologies. Since the size of the data samples is constantly increasing [2, 3], Reactome offers a set of pathway analysis tools which aim to deal with this scenario and yet provide reliable and accurate results with interactive (seconds) response time for genome-wide datasets.

* Correspondence: hhe@ebi.ac.uk

¹European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, UK

¹⁰State Key Laboratory of Proteomics, Beijing Proteome Research Center, Beijing Institute of Radiation Medicine; National Center for Protein Sciences, 102206, Beijing, China

Full list of author information is available at the end of the article



© The Author(s). 2017 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated.

Here, we are discussing the high performance Reactome implementation of the well established over-representation analysis (ORA) method [4], focussing on the computer science aspect, elaborating on the different data structures and design patterns used to optimise the execution time and reduce the server load.

Initially the focus is on the strengths and weaknesses of keeping the data directly in a relational database and its usage to perform in-database analyses. Then we continue with a detailed explanation of the new pathway analysis approach, and conclude with the presentation of the results and the discussion.

The relational databases approach

Relational databases are widely used in pathway knowledge-bases for data management; either during curation, the release process or in the final production phase. It is also very common to store the information in third normal form due to its convenience for data integrity assurance [5–7].

Relational databases in their third normal form can be efficient in computational terms. For the above mentioned use cases, however, this approach greatly slows the execution of analysis algorithms, due to the size of the temporary tables for the queries and later projections. For this reason database-based analysis approaches use denormalised versions of the databases instead [8]. The denormalisation process replicates a lot of data to speed up the queries but it may penalise analysis execution time as the original database content grows bigger.

Focusing on the computational side of the problem, the query containment problem is undecidable for relational algebra and SQL, but is decidable and NP-complete for conjunctive queries. In fact, the query containment problem for conjunctive queries is exactly the same problem as the query evaluation problem [9]. When queries tend to be small, NP-completeness is usually considered acceptable but its performance falls when queries tend to be big. In addition, it is also worth considering that creating intermediate tables in memory after executing a “join” statement is one of the heaviest operations for a database engine.

Reactome’s previous implementation of the pathway analysis was based on a denormalised version of the Reactome relational database. Among its limitations were that it provided results only of the higher-level pathways in Reactome, and the lack of programmatic access. In addition, the previous implementation suffered from poor performance mainly due to the fact that, on every analysis request, it connected to the relational database, rather than querying an intermediate in-memory data structure. Thus, the response time of the previous Reactome analysis could reach 5 min, as soon as the user sample included a few hundreds of gene

identifiers, causing a high server load that, combined with a number of concurrent analysis requests, affected the stability of the Reactome website and often resulted in outages.

In resources like Reactome, analyses use not only curated data but also extra information and cross-references to other resources that are included in the final version of the database, for example to allow usage of identifiers from other resources than the main ones used by the curators to identify proteins, genes, micro-RNAs or chemicals. Each major resource uses its own conventions when assigning identifiers, so the problem of mapping the various, potentially unstable, identifiers that refer to identical entities, commonly known as identifiers mapping, constitutes a major challenge. There is a number of resources that aim to provide a solution to this problem, most notably, the Protein Identifier Cross-Reference (PICR) [10], BridgeDB [11] and UniProt [12]. However, Reactome addresses this problem during each release process by cross-referencing every curated entity to other resources. In particular, based on the UniProt or ChEBI identifiers of the curated entities, filled in during curation, Reactome queries Orphanet, Protein Ontology (PRO), IntAct, RHEA, DOCKBlaster, FlyBase, The Human Metabolome Database (HMDB), Zinc, KEGG, UniProt, ENSEMBL, BRENDA and IntEnz to get their cross-references for entities annotated in Reactome. Both, curated and cross-referenced identifiers are included in the analysis lookup table, as explained in the Implementation section.

As the amount of curated data in Reactome grows and the number of cross-references increases due to the inclusion of new resources, the database-based approach does not scale well, so there is a need to implement a new approach to provide fast, accurate and reliable analysis tools to the final users. This new approach is based on the concatenation of different steps, each one resolved via the appropriate data structure, as explained in the next section.

Implementation

Identifying a convenient data structure to solve a given problem is one of the main factors to achieve a high performance final product. As Skiena explains in [13], picking the wrong data structure for the job can be disastrous in terms of performance but identifying the very best data structure is usually not as critical, because there can be several choices that perform similarly.

Based on the divide and conquer rule, the first step is breaking down the analysis problem into different sub-problems simple enough to be solved in polynomial time by identifying a convenient data structure. Here, the analysis algorithm can be split into four parts: (1) checking whether the user’s protein/chemical identifiers are

present in Reactome, (2) for the present ones, finding whether these are parts of complexes and/or sets as well as the species projection, (3) aggregating the found identifiers in the pathways (and super-pathways) where these are present and finally (4) performing the statistical testing to calculate the likelihood that the association between the sample identifiers and the found pathway is due to random chance.

Further on in this section each part is discussed in detail to determine its peculiarities; to expose the chosen data structure and the mechanisms adopted for its improvement; and to show how to connect each step to the following one to come up with the final improved analysis algorithm. Another point of emphasis for optimisation will be the memory usage of each step, so that the filled data structures can be kept in memory to improve the performance of the data traversing algorithms implemented on top of them.

User sample identifiers search in Reactome

Annotated physical entities (PE) in Reactome can be either single entities or complexes. Single entities include proteins, small molecules, RNA, DNA, carbohydrates, or lipids, whilst complexes consist of a combination of any of the single entities, or polymers synthesized from the single entities. However, apart from these two main categories, curators in Reactome can group related entities into sets. PEs are the building blocks that later on will be used as inputs, outputs, catalysts or regulators in reactions.

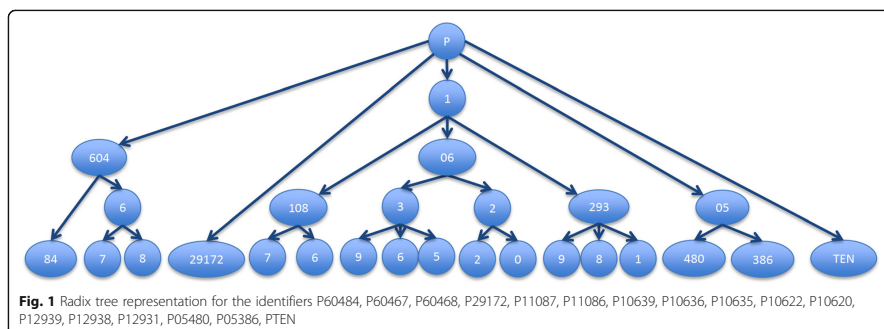
Identifiers or accession numbers are used to unequivocally refer to a single entity, but PEs have different slots to hold the main identifier, secondary identifier, cross-references, synonyms and other identifiers. The main identifier slot is always manually annotated by the experts who curate data in Reactome (curators), and the other slots can either be manually filled during curation or automatically populated during the release process. This strategy allows storing identifiers for a wide range of

resources: UniProt, ChEBI, Ensembl, miRBase, GenBank/EMBL/DBJ, RefPep, RefSeq, EntrezGene, OMIM, InterPro, Affymetrix, Agilent, KEGG Compound, Illumina, etc.

Therefore, in the first part of the analysis, the main requirement is to improve the process of finding out whether each identifier in the user's sample corresponds to one or many PEs in Reactome. An identifier corresponds to a PE if it matches with any of the identifiers stored in the different slots mentioned afore. In fact, the best way to solve this problem is by following the reverse approach; creating a lookup table with all the corresponding PEs per each identifier cross-referenced in Reactome. As a consequence, another important requirement is to minimise the memory usage so the data can be kept in memory to improve the query time.

The selection of a good data structure is then determined by requirements both to implement a fast lookup table and to keep memory usage low. A Trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings [14]. A radix tree is a space-optimized Trie data structure where each node with only one child is merged with its parent [15].

On the one hand, a radix tree has relatively low memory usage for the lookup table because the common prefixes are shared avoiding data duplication (Fig. 1). On the other hand, the cost of comparing a search key for equality with a key from the data structure can be a dominant cost which cannot be neglected. The radix tree string lookup algorithm fits the analysis algorithm's original purpose because iterating over tree nodes keeps the identifier seeking time restricted to each identifier's length and existence in the Reactome target set. As a consequence of this, in case the searched identifier is not contained in the data structure, there is no need to read all of it as happens in the hashing methods where the hash value of the string has to be calculated in every case by reading it entirely.



In summary, once a tree node is reached following the radix tree lookup algorithm for a given identifier, the presence or absence of references to PEs indicates whether the associated identifier is present or not in the database. Actually, the mentioned "references to PE" are indeed pointers to nodes in the data structure chosen for the next part of the analysis.

Reactome uses unique primary identifiers for the PEs it references, in particular UniProt for proteins and ChEBI for chemical entities. Thus, if users submit datasets using these reference systems, the mapping to PEs is straightforward. However, following frequent user requests, we also accept input data with non-unique identifiers, in particular gene names. These are then potentially mapped to multiple PEs. Thus, each target node in the tree could contain more than one pointer to the next data structure.

Traversing complexes/sets composition and species projection

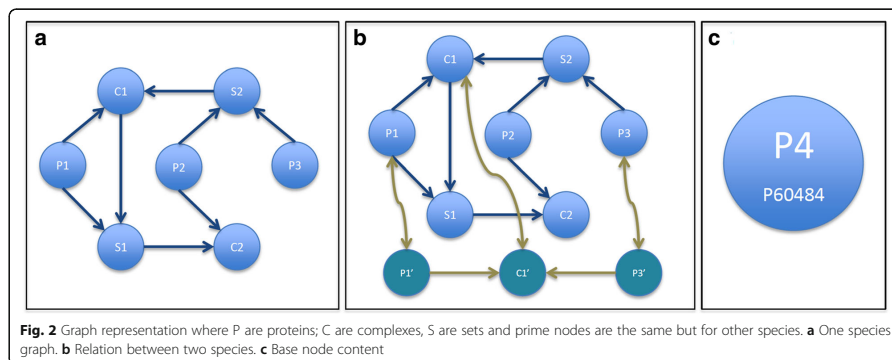
Reaching the associated single entity for a given identifier is the beginning of the second step in the analysis. When these single entities are part of a complex, they are also a target in this step of the analysis. Besides the single entities and complexes, there is another type of PE called sets which, along with complexes, are also to be considered. A set is an abstract representation of a group of two or more entities which are not interacting with each other but are functionally equivalent in the situation where the set is used, for example multiple members of a family of enzymes that could each potentially catalyse a reaction. Furthermore, complexes and sets can also contain other complexes and sets in order to represent much more elaborate structures causing the problem's intricacy to grow.

Another specific requirement is the possibility of performing species projection to collect the results for *Homo sapiens* independently of the species for which the identifiers are provided, to benefit from the more complete Reactome annotation for Human. To do so, the species orthologs annotated in Reactome have to be taken into account. Orthologs are entities in different species that evolved from a common ancestor by speciation.

The last requirement in this step is to keep track of the identifiers mapping between the submitted identifiers and those used in Reactome to curate the single entities: UniProt accessions for proteins, Ensembl identifier for genes, ChEBI identifiers for small molecules and miRBase for microRNAs. Although an important part of this mapping started by including the known cross-references as identifiers in the radix tree in the previous step, the mapping itself has to be implemented in this step.

Summarising the exposed requirements for this step of the analysis, the chosen data structure has to model the entities composition problem, the species orthologs projection and the entities mapping. A directed graph is a graph, or set of nodes connected by edges, where the edges have a direction associated with them. For a given graph G with several nodes (a , b , c and d), if G has an arrow from a to b and another arrow from b to c , then the composed graph G^2 has an arrow from a to c . If G has an arrow from a to b , another arrow from b to c and yet another from c to d , then the composed graph G^3 has an arrow from a to d .

Building one graph per species (Fig. 2a) and interconnecting all of them linking all the ortholog nodes (Fig. 2b) creates a bigger graph where the projection requirement is then satisfied. Due to the node uniqueness



in the final graph, for those cases where a node is part of one or more structured entities, it contains as many edges pointing to other graph nodes as structures in which it is contained, so structured entities are easily modelled. Finally, if each node of the graph contains its associated entity main identifier (Fig. 2c), when it is reached from a radix tree node representing an identifier other than the main one, this association is stored in order to be offered as part of the result as the required mapping once the analysis is finished.

The graph in Fig. 2a shows three proteins (P1, P2 and P3), two complexes (C1 and C2), and two sets (S1 and S2). By following the edge from node to node, S2 could be either P2 or P3, formally represented as {P2,P3}. C1 is a complex which, due to its edge from S2, is then potentially two complexes: {P1,P2} or {P1,P3}, represented as {{P1,P2},{P1,P3}}. Following this deconstruction, S1 is then [P1, {P1,P2}, {P1,P3}] and finally C2 is {{P1,P2}, {{P1,P2},P2}, {{P1,P3},P2}}.

For instance, when an identifier matching with P3 is processed and its corresponding node in the graph is reached from the radix tree, it takes miniscule processing time to traverse the graph and reach the nodes S2, C1, S1 and C2. Likewise, if the target protein is P1, the reachable nodes following the graph edges are C1, S1 and C2. In both examples each target protein is part of the complexes and sets represented by the traversed nodes.

Employing a graph improves the analysis algorithm cost and, important in building an in-memory analysis, the memory usage is kept low because there is no data duplication as the node for a given main identifier is only in memory once. In addition, the final number of node iterations of the algorithm is limited by the related entities for a given identifier, avoiding queries against a large amount of data and intermediate results merging, as done in the database based approach.

As for the radix tree described above, the graph also requires a strategy to allow the algorithm to move on to the next analysis step. In this case, each graph node representing an entity directly associated to one or several pathways will contain as many links to the following data structure as different locations where it is present. Although in the current analysis step each entity associated with the target identifier is found, for the final result and the statistics calculation, there is still one more data structure to be used, as explained in the following sub-section.

Results aggregation into the pathways organisation

Every PE that was directly or indirectly hit in the previous step is associated to one or more pathways. To calculate the significance of each pathway, for a given user sample, it is essential to determine the number of entities found per pathway. Due to the parent-child organisation of the Reactome pathways in an ontology-like hierarchy, when an entity is present in a certain pathway it is also present in its super-pathways in a recursive manner until a top-level pathway is reached (i.e. if a protein is present in "Metabolism of carbohydrates", it is also present in "Metabolism").

Taking into account the requirements previously discussed, a good data structure to model this step is a double-linked tree, where each node represents a pathway and contains links to its parent and children (Fig. 3). When a node in the tree is hit, the action can be recursively propagated all the way up to the root. To reduce the memory footprint only identifiers, names and placeholders for results calculation are kept in each node.

Apart from being a convenient data structure to speed up collection of results and a good holder for the statistics results, once the analysis is finished, this data structure can also be serialised to a file to persist the result.

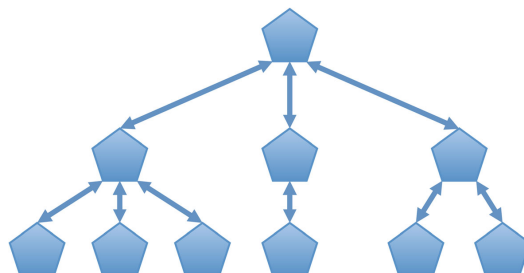


Fig. 3 Double-linked tree to represent the event hierarchy in Reactome. The root node defines the species and its children represent the different pathways and sub-pathways in Reactome. Each node contains the pathway identifier, name, the total curated entities and the number of entities found in the user's sample

In addition, associating the file to a token provides an easy way to create finer grained methods that allow filtering of the result on the server side to help speeding up light-weight clients. In this scenario, the clients can keep the token once the initial analysis is finished and depending on the user's needs, perform several requests to the server referencing the associated token.

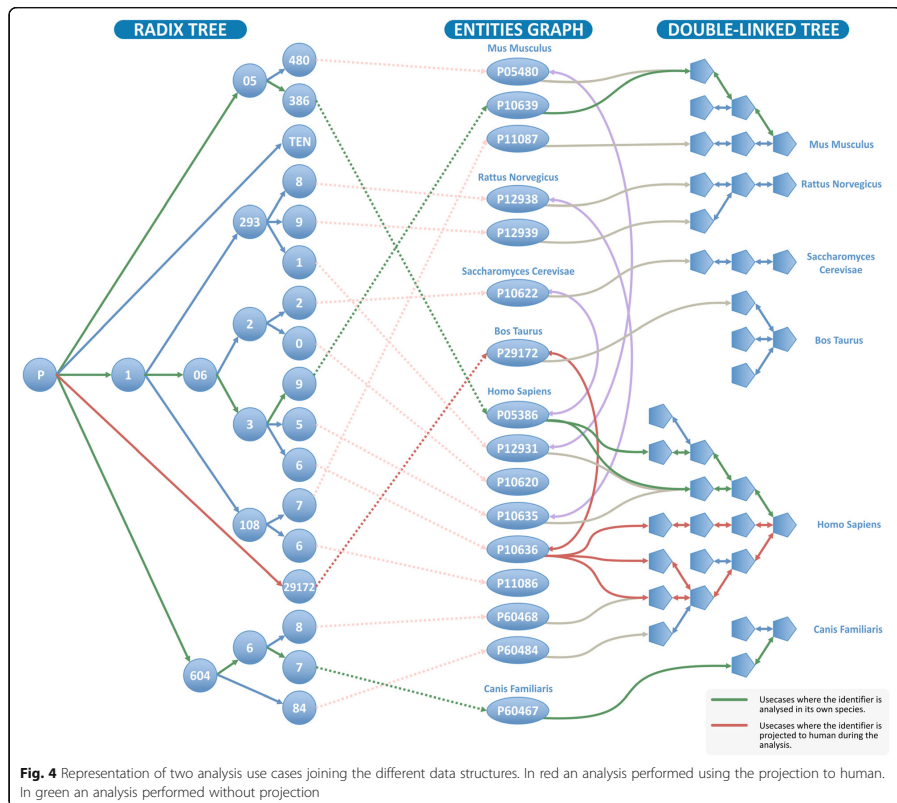
Analysis result statistics calculation

The basic hypothesis in an over-representation analysis is that relevant pathways can be detected if the proportion of differentially expressed genes, within a given pathway, exceeds the proportion of genes that could be randomly expected [1]. Consequently, the fourth and last step in the analysis method involves the statistics calculation. This step does not require any extra data structure because the double-linked tree fits perfectly to the purpose.

The *p*-Value shows the statistical significance of each hit pathway for a given sample and the background for which the analysis has been performed. In Reactome the method used to calculate the statistical significance is the Binomial Test. Together with the *p*-Value, the False Discovery Rate (FDR) helps estimating the false positives and it is calculated using the Benjamini-Hochberg approach [16]. As mentioned afore, we have focussed on optimising the performance of the Reactome pathway analysis, while maintaining the basic algorithm as previously published [17].

Results and discussion

This paper shows how splitting the pathway analysis method in four steps, in a way that every challenge can be easily addressed in a polynomial time using the appropriate data structures, speeds up the process and minimises the memory usage so the whole data structure can be kept



in memory for a high-performance analysis. The result is a new set of analysis tools which vastly improve Reactome analysis interface performance and stability.

Summarising the steps (Fig. 4), for each identifier in the user's sample, the first action is to find whether it is present in Reactome using a previously built radix-tree as a look-up-table. This speeds up the process, keeping a low memory footprint. For those that are present, the radix-tree nodes point to one or many nodes in a graph which is used as the second data structure to keep the curated relations between PEs as well as species orthology. Traversing this second data structure, applying or not the projection to species, provides pointers to all pathways stored in the final data structure, which is a double-linked tree, that helps aggregating the result and acts as a placeholder for the last step when the analysis statistics are calculated.

The described method has been developed using Java as programming language and can be downloaded from <https://github.com/reactome/AnalysisTools>. This package contains two main modules; *Core* and *Service*. The improved strategy has been developed in the *Core*, where the analysis is executed. The *Service* module is a Spring MVC (<http://spring.io/>) layer to create a RESTful service with a documented API, using OpenAPI, formerly known as Swagger v2.0 (<http://swagger.io/>), providing programmatic access. Hence, there are two ways of accessing the analysis tools; (1) programmatically via a web service (<http://reactome.org/AnalysisService/>) or (2) through a graphical user interface directly integrated in Reactome's Pathway Browser (<http://reactome.org/PathwayBrowser/#/TOOL=AT>).

The web service is used to integrate the analysis in other system's scripts, pipelines or to integrate the analysis in third-party applications. More information on how to do so can be found in Reactome's developer zone (<http://goo.gl/k5ffhu>).

The pathway analysis approach described here is deployed in the Reactome production web site, stably handling on average 10.850 analysis requests from 2.000 unique users per month in the first half of 2016. Memory usage for the Apache Tomcat running this service plus other services in the server side is set to 2GB.

Comparison with other resources

Among the plethora of pathway databases [1], there are resources with similar tools that perform overrepresentation analysis. Most notably, Gene Set Enrichment Analysis (GSEA) [18], the Database for Annotation, Visualization and Integrated Discovery (DAVID) [19], the Protein Analysis Through Evolutionary Relationships (PANTHER) [20] and ConsensusPathDB [21] are using similar statistical algorithms in their implementations and are freely available for academic use. Table 1 presents a comparison among these resources. For the comparison of processing time, only the first column in the four test sets, containing the gene identifiers, has been used. Reactome uses all genes annotated in the knowledge base as the background distribution. To our knowledge, this is also the approach used in the comparator tools, and we have not used options for custom background distributions, as statistics calculation could take longer in this scenario.

Table 1 Comparison of resources providing analysis methods and accessibility

Resource	Analysis methods	Online tool	Programmatic access	Processing time				
				Hippocampal atrophy - 79 genes ^a	Migraine disorder - 644 genes ^b	Parkinson's disease - 1492 genes ^c	Multiple sclerosis - 2570 genes ^d	Inflammatory bowel disease - 4110 genes ^e
PANTHER	ORA	✓	-	~2 s	~4 s	~6 s	~8 s	~12 s
Consensus PathDB	ORA	✓	SOAP/WSDL	~1 min	~1 min	~3 min	~3 min	~1 min
DAVID	ORA	✓	SOAP/WSDL	~4 s	~4 s for conversion of official gene ids to 7498 DAVID ids. Analysis not performed - sample size limitation	~5 s for conversion of official gene ids to 17272 DAVID ids. Analysis not performed - sample size limitation	~8 s for conversion of official gene ids to 29420 DAVID ids. Analysis not performed - sample size limitation	Not performed - sample size limitation
GSEA	ORA	-	-	-	-	-	-	-
REACTOME v1.0	ORA	✓	-	~2 min	~7 min	~12 min	~19 min	~25 min
REACTOME v2.0	ORA	✓	REST	~1 s	~1 s	~2 s	~2 s	~3 s

Comparison between different resources and whether they provide analysis methods which are accessible online (UX or programmatic access) and the average response time for a predefined sample. For the comparison of processing time, only the first column in the test sets -the gene identifiers- has been used. Datasets are available in

^ahttps://www.targetvalidation.org/disease/EFO_0005039/associations (accessed 13/07/2016)

^bhttps://www.targetvalidation.org/disease/EFO_0003821/associations (accessed 13/07/2016)

^chttps://www.targetvalidation.org/disease/EFO_0002508/associations (accessed 13/07/2016)

^dhttps://www.targetvalidation.org/disease/EFO_0003885/associations (accessed 13/07/2016)

^ehttps://www.targetvalidation.org/disease/EFO_0003767/associations (accessed 13/07/2016)

GSEA offers its analysis tool exclusively through a desktop application and therefore requires download and installation before usage, rendering the tool suitable for more experienced users. On the other hand, DAVID, PANTHER and ConsensusPathDB provide online access to their analysis tools via a web interface, similarly to REACTOME. Thus, users can submit their sample for analysis through their favourite web browser.

Furthermore, besides REACTOME, DAVID and ConsensusPathDB are also allowing users to access their analysis tools programmatically, through a set of web services. Hence, researchers and software developers can integrate the provided analysis tools into their pipelines and applications. However, while DAVID and ConsensusPathDB rely on the Simple Object Access Protocol (SOAP) and the Web Service Description Language (WSDL) for their web services, Reactome analysis web service is based on the Representational State Transfer (REST). The adoption of REST eliminates the need for complex clients and renders Reactome analysis service simpler, more lightweight, more flexible, and, thus, easier to integrate into third party software compared to its SOAP/WSDL counterparts.

Leveraging on the performance gained by the in-memory analysis approach explained above and the use of RESTful web services, the Reactome analysis tool does not impose any limitations on the sample size or the frequency of analysis requests, unlike DAVID. Regarding its weaknesses compared to DAVID, Reactome analysis tool has a more limited coverage, as it does not integrate as many resources as DAVID does, but it focuses on high quality manually curated pathways that are updated quarterly. In addition, Reactome does not allow users to customise the background population of their analysis.

Conclusions

Through the use of highly optimised, in-memory data structures and algorithms, Reactome has achieved a stable, high performance pathway analysis service, enabling the analysis of genome-wide datasets within seconds, allowing interactive exploration and analysis of high throughput data.

Availability and requirements

All data generated or analysed during this study are included in this published article.

Source code: <https://github.com/reactome/AnalysisTools>

Web service: <http://reactome.org/AnalysisService/>

User interface: <http://reactome.org/PathwayBrowser/#/TOOL=AT>

Documentation: <http://goo.gl/k5ffhu>

Abbreviations

API: Application program interface; DNA: Deoxyribonucleic acid; FDR: False discovery rate; MVC: Model view controller; NP: Non-polynomial; ORA: Over-representation analysis; PE: Physical entities; REST: Representational State Transfer; RNA: Ribonucleic acid; SOAP: Simple object access protocol; SQL: Structured query language; WSDL: Web Service Description Language

Acknowledgements

We thank Pablo Porras for his valuable contribution in the requirements definition.

Funding

National Institutes of Health BD2K grant [U54 GM114833]; European Bioinformatics Institute (EMBL-EBI); Open Targets (The target validation platform); National Human Genome Research Institute at the National Institutes of Health [U41 HG003751]; Ontario Research (GL2) Fund. Funding for open access charge: National Institutes of Health [U54 GM114833]. The funding bodies had no role in the design or conclusions of the study.

Authors' contributions

AF designed and implemented the proposed analysis method and was the major contributor in writing the manuscript. KS provided technical support and contributed in the writing. GV developed the input parser. OF offered technical support and developed the part of the service related to data storage and management. PM, VA, PD, LS and HH provided scientific support and helped with the validation of the results. PM, PD and HH also contributed in the writing. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Author details

¹European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, UK. ²Open Targets, Wellcome Genome Campus, Hinxton, UK. ³Fundación Investigación INCLIVA, Universitat de València, Valencia, Spain. ⁴Instituto de Medicina Genómica, Valencia, Spain. ⁵Escuela Técnica Superior de Ingenierías, Universitat de València, Valencia, Spain. ⁶Institute for Integrative Systems Biology (I2SysBio), Universitat de València-CSIC, Paterna, Valencia, Spain. ⁷NYU Langone Medical Center, New York, USA. ⁸Ontario Institute for Cancer Research, Toronto, Canada. ⁹Department of Molecular Genetics, University of Toronto, Toronto, Canada. ¹⁰State Key Laboratory of Proteomics, Beijing Proteome Research Center, Beijing Institute of Radiation Medicine; National Center for Protein Sciences, 102206, Beijing, China.

Received: 14 July 2016 Accepted: 22 February 2017

Published online: 02 March 2017

References

- García-Campos MA, Espinal-Enríquez J, Hernández-Lemus E. Pathway analysis: state of the art. *Front Physiol*. 2015;6:383.
- Zhang J, Chioldini R, Badr A, Zhang G. The impact of next-generation sequencing on genomics. *J Genet Genomics*. 2011;38:95–109.
- Reuter JA, Spacek DV, Snyder MP. High-throughput sequencing technologies. *Mol Cell*. 2015;58(4):586–97.
- Drăghici S, Khatri P, Martins RP, Ostermeier GC, Krawetz SA. Global functional profiling of gene expression. *Genomics*. 2003;81:98–104.
- Chowdhury S, Sarkar RR. Comparison of human cell signaling pathway databases—evolution, drawbacks and challenges. *Database* (Oxford). 2015. doi:10.1093/database/bau126.
- Shin SK, Sanders GL. Denormalization strategies for data retrieval from data warehouses. *Decis Support Syst*. 2006;42(1):267–82.
- Codd EF. In: Rustin R, editor. Further normalization of the data base relational model, data base systems. Englewood Cliffs: Prentice-Hall; 1972.
- Talbi E, Zomaya AY. Grid computing for bioinformatics and computational biology. Hoboken: Wiley-Interscience; 2008.

9. Abiteboul S, Hull RB, Vianu V. Foundations of databases: the logical level 1st. Boston: Addison-Wesley; 1995.
10. Cote RG, Jones P, Martens L, Kerrien S, Reisinger F, Lin Q, Leinonen R, Apweiler R, Hermjakob H. The Protein Identifier Cross-Referencing (PICR) service: reconciling protein identifiers across multiple source databases. *BMC Bioinformatics*. 2007;8:401.
11. Van Iersel MP, Pico AR, Kelder T, Gao J, Ho I, Hanspers K, et al. The BridgeDb framework: standardized access to gene, protein and metabolite identifier mapping services. *BMC Bioinformatics*. 2010;11:5. doi:10.1186/1471-2105-11-5.
12. UniProt Consortium. UniProt: a hub for protein information. *Nucleic Acids Res*. 2014;43:D204–12.
13. Skiena SS. The algorithm design manual. London: Springer; 2008.
14. De la Briandais R. File searching using variable length keys. Proceedings of the Western Joint Computer Conference. 1959; 295–298.
15. Morrison D. PATRICIA-Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J ACM*. 1968;15(4):514–34.
16. Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J Roy Statist Soc Ser B*. 1995;57:289–300.
17. Wu G, Dawson E, Duong A, Haw R, Stein L. ReactomeFIViz: a cytoscape app for pathway and network-based data analysis. *F1000Research*. 2014;3:146.
18. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci*. 2005;102:15545–50.
19. Huang DW, Sherman BT, Lempicki RA. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat Protoc*. 2009;4(1):44–57.
20. Mi H, Poudel S, Muruganujan A, Casagrande JT, Thomas PD. PANTHER version 10: expanded protein families and functions, and analysis tools. *Nucleic Acids Res*. 2016;44:D336–42.
21. Kamburov A, Pentchev K, Galicka H, Wierling C, Lehrach H, Herwig R. ConsensusPathDB: toward a more complete picture of cell biology. *Nucleic Acids Res*. 2011;39:D712–7.

Submit your next manuscript to BioMed Central
and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at
www.biomedcentral.com/submit



P.2. REACTOME DIAGRAM VIEWER: DATA STRUCTURES AND STRATEGIES TO BOOST PERFORMANCE

Bioinformatics, 34(7), 2018, 1208–1214
doi: 10.1093/bioinformatics/btx752
Advance Access Publication Date: 23 November 2017
Original Paper

OXFORD

Databases and ontologies

Reactome diagram viewer: data structures and strategies to boost performance

Antonio Fabregat^{1,2,*}, Konstantinos Sidiropoulos¹, Guilherme Viteri¹, Pablo Marin-Garcia^{3,4}, Peipei Ping⁵, Lincoln Stein^{6,7}, Peter D'Eustachio⁸ and Henning Hermjakob^{1,9,*}

¹European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton CB10 1SD, UK, ²Open Targets, Wellcome Genome Campus, Hinxton CB10 1SD, UK, ³Fundación Investigación INCLIVA, Universitat de València, Valencia, Spain, ⁴Instituto de Medicina Genómica, Valencia, Spain, ⁵NIH BD2K Center of Excellence and Department of Physiology, Medicine and Bioinformatics, University of California, Los Angeles, CA 90095, USA, ⁶Ontario Institute for Cancer Research, Toronto ON M5G 0A3, Canada, ⁷Department of Molecular Genetics, University of Toronto, Toronto ON M5G 0A3, Canada, ⁸NYU Langone Medical Center, New York NY 10016, USA and ⁹State Key Laboratory of Proteomics, Beijing Proteome Research Center, Beijing Institute of Radiation Medicine, National Center for Protein Sciences, Beijing 102206, China

*To whom correspondence should be addressed.

Associate Editor: Janet Kelso

Received on May 19, 2017; revised on November 2, 2017; editorial decision on November 15, 2017; accepted on November 22, 2017

Abstract

Motivation: Reactome is a free, open-source, open-data, curated and peer-reviewed knowledge-base of biomolecular pathways. For web-based pathway visualization, Reactome uses a custom pathway diagram viewer that has been evolved over the past years. Here, we present comprehensive enhancements in usability and performance based on extensive usability testing sessions and technology developments, aiming to optimize the viewer towards the needs of the community.

Results: The pathway diagram viewer version 3 achieves consistently better performance, loading and rendering of 97% of the diagrams in Reactome in less than 1 s. Combining the multi-layer html5 canvas strategy with a space partitioning data structure minimizes CPU workload, enabling the introduction of new features that further enhance user experience. Through the use of highly optimized data structures and algorithms, Reactome has boosted the performance and usability of the new pathway diagram viewer, providing a robust, scalable and easy-to-integrate solution to pathway visualization. As graph-based visualization of complex data is a frequent challenge in bioinformatics, many of the individual strategies presented here are applicable to a wide range of web-based bioinformatics resources.

Availability and implementation: Reactome is available online at: <https://reactome.org>. The diagram viewer is part of the Reactome pathway browser (<https://reactome.org/PathwayBrowser/>) and also available as a stand-alone widget at: <https://reactome.org/dev/diagram/>. The source code is freely available at: <https://github.com/reactome-pwp/diagram>.

Contact: fabregat@ebi.ac.uk or hhe@ebi.ac.uk

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

Reactome (<https://reactome.org>) is a free, open-source, open-data, curated and peer-reviewed knowledgebase of biomolecular pathways. It provides bioinformatics tools for visualization, interpretation and analysis of biomolecular data to support basic research, genome analysis, modelling, systems biology and education.

At the cellular level, life is a network of molecular reactions that include signal transduction, transport, DNA replication, protein synthesis and intermediary metabolism. In Reactome, these processes are systematically described in molecular detail to generate an ordered network of molecular transformations, resulting in an extended version of a classic metabolic map described by a single, consistent data model (Fabregat *et al.*, 2016). The Reactome knowledgebase thus systematically links human proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for exploring and discovering unexpected functional relationships in data such as gene expression pattern surveys or somatic mutation catalogues from tumour cells. In Reactome the steps of a pathway are represented as connected molecular events termed 'reactions'. Reactome's content is organized into a set of canonical pathways that corresponds to distinct biological processes with minimal overlap of reactions and proteins, arranged in a hierarchy corresponding to the GO biological process hierarchy. Each pathway is represented in a pathway diagram laid out following the Systems Biology Graphical Notation (SBGN) (Le Novère *et al.*, 2009) process description language (Fabregat *et al.*, 2016). Additionally, Reactome offers a pathway analysis service that supports enrichment and expression analysis (Fabregat *et al.*, 2016, 2017). Users can submit their own dataset for analysis and visualize the result as overlays on top of pathway diagrams.

Web browsers are one of the main types of application used for retrieving, presenting and traversing information resources on the World Wide Web. Creating an interactive pathway diagram viewer for web browsers poses a series of challenges that need to be addressed in order to offer a fast-loading and responsive product. On the one hand, implementing a custom solution enables full control over features and capabilities at the cost of longer development time. On the other hand, reusing existing software has the advantage of launching the final product in a shorter period of time but with additional features limited by the existing capabilities of the selected third party software (Krueger, 1992). Some resources like MINERVA (Gawron *et al.*, 2016) and NAVICELL (Kuperstein *et al.*, 2013) have adopted the Google mapTM engine. Others such as Pathway Commons (Cerami *et al.*, 2011), WikiPathways (Kutmon *et al.*, 2016) and KEGG (Kanehisa *et al.*, 2014) developed and use their own viewers.

Reactome has always used an in-house developed diagram viewer which has evolved over the years to include enhancements in usability and performance as a response to extensive usability testing sessions aiming to improve the tool towards the needs of the community (Roto *et al.*, 2009). When the second version of the diagram viewer was released in 2013, systematic user experience testing and informal user feedback pointed out that the loading time and user interactivity needed to be improved. We describe here how we addressed these challenges, by implementing a more efficient diagram storage format, and by adopting new strategies for client data storage, retrieval and rendering. Additionally, this study aims to provide guidance to other researchers or groups working on similar visualization tools.

2 Implementation

The usability testing sessions showed that the users (i) had trouble using the diagram search functionality, (ii) found the diagrams too crowded/complex, especially in zoomed-out views and (iii) often lost diagram context while navigating through the event hierarchy due to the diagram's flashing and abrupt changes of location, instead of an animated transition to the target position. Other comments highlighted the fact that the zoom was not progressive, but instead users could only zoom in predefined steps.

Aiming to address these challenges and enhance the overall user experience, a new version of the Pathway Diagram Viewer was implemented. The new version (version 3) was also focused on faster data loading, diagram rendering and element seeking. This decision was made based on the fact that users retain the feeling of being in control when an interaction between them and the computer takes no more than one second (<http://www.nngroup.com/articles/powers-of-10-time-scales-in-ux>).

Improvements targeted different levels and included: (i) restructuring of the data format used to send the data from the server to the client, (ii) using a graph data structure to store the pathway content on the client side, (iii) boosting the client content load strategy, (iv) implementing a multi-layer canvas approach, (v) utilising a space partitioning data structure to store the elements to be rendered and (vi) employing the delegate design pattern to control the flow of information based on the level of zoom. This section delves deeper into each aspect to describe them in finer grain.

2.1 Data format update

The first step to improve the overall user experience was to reduce the client loading time by replacing the eXtensible Markup Language (XML) format (<https://www.w3.org/TR/REC-xml>) for diagram data storage with JavaScript Object Notation (JSON) (<http://www.json.org>). JSON is less verbose than XML and thus has a smaller footprint. More important, JSON's natural mapping to JavaScript objects is faster and uses fewer resources than its XML counterpart (Boci *et al.*, 2012; Nurseitov *et al.*, 2009; Wang, 2011). For all these reasons, resources that rely heavily on XML for their storage format, could potentially benefit from transitioning to JSON.

Therefore, all Reactome pathways containing diagram layout information are converted from XML to JSON and stored on the server side as static resources during the quarterly release process. In the same process, for every diagram, a graph of all the contained entities and reactions is generated and stored in an additional JSON file to enable a richer browsing and search experience throughout the diagram content. The next subsection elaborates on the creation of the graph and its usage along with the layout information.

2.2 Underlying graph structure

Among other elements, diagrams contain macromolecular complexes and entity sets comprised of components and members, respectively. Entity sets are used to group entities together based on common properties. Sets and complexes may have other complexes or sets as their constituents (D'eustachio, 2011). This approach quickly builds up to a highly structured network of contained entities that, in most diagrams, is conveniently represented by a single glyph that simplifies the view. Thus, the diagram viewer must be aware of all this information and able to take full advantage of it, in order to provide a much richer search function and smarter interaction with the constituents of complexes/sets.

For example, a search for a protein should highlight not only instances of the protein visible in the diagram but also any complex

1210

A.Fabregat et al.

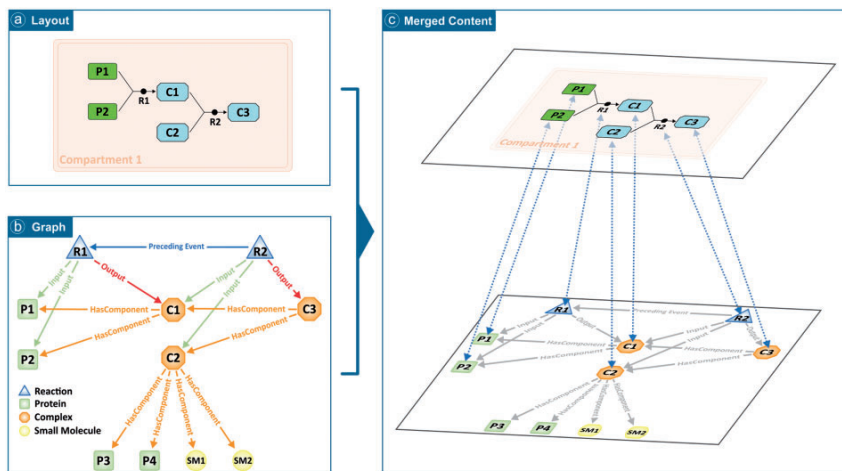


Fig. 1. Schematic view of a pathway made up of two reactions. (a) The pathway diagram as presented to the final user. (b) Underlying graph with the whole content of the pathway. (c) Representation of the merging of both the diagram and graph on the client side. In the figure, P_n are proteins, SM_n are chemicals, C_n are complexes and R_n are reactions. From the graph, it can be extracted that C_1 contains $[P_1, P_2]$, C_2 contains $[P_3, P_4, SM_1, SM_2]$ and C_3 contains $[C_1, C_2]$, but by traversing the graph it can be easily inferred that C_3 actually contains $[P_1, P_2, P_3, P_4, SM_1, SM_2]$

instances of which the protein is a part and any set of which it is a member.

In previous versions, the client retrieved a file with the identifiers defining each element present in the diagram from the server side. In the new version, a file with a graph representing the content of the different complexes and sets for each diagram and annotating the participants of every included reaction is required (Fig. 1). This approach introduced an additional file with the graph content that has to be consumed separately by the client and merged with the layout data, once both are loaded.

The graph and layout content have elements in common, but in most cases the graph will contain more information. In the example presented in Figure 1, the pathway diagram layout contains 7 elements; 5 entities and 2 reactions (Fig. 1a), and the graph contains 11 elements; 9 entities and 2 reactions (Fig. 1b). The 4 extra elements in the graph can be justified by the fact that none of the components of C_2 (4 entities) are present in the layout. Another benefit of the graph is that entities that are part of different complexes or sets are represented only once and remain accessible via graph traversing.

Because of the complementary nature of information stored in the layout and the graph files, the client side needs to implement a technique that merges both contents and allows them to seamlessly work together (Fig. 1c). Our approach is to propagate user actions from the layout level down to the graph level in order to have an easy way to traverse the content and identify the relevant entities to be highlighted by traversing up to the layout again. In addition, the built-in search feature can now take into account not only entities that are represented by a glyph in a diagram, but also all the contained entities composing that glyph. For instance, users can search among all components/members of the complexes/sets present in a single diagram. The client is able to highlight all those diagram entities containing a component (or member) that matches the search term.

For most applications that feature interactive visualizations, accompanying layout information with additional semantic metadata can prove a good practice, as it enriches the visualizations by assigning a meaning to all visual entities. In addition, this extra information can be used to enrich any existing search functionality by extending it to more than what is visualized.

2.3 Updated loading and caching strategies

The introduction of separate layout and graph files was accompanied by the adoption of a render-first loading strategy in the client (Fig. 2). The client makes concurrent XMLHttpRequest calls for the layout and the graph data content (<https://xhr.spec.whatwg.org>). As soon as the layout data is available, the viewer processes it and renders the diagram on the canvas. Once the graph content is ready, the latter is processed and linked to the diagram layout to be used for interactive navigation, search and future analysis overlay purposes. Following this render-first approach, the new version of the diagram viewer primes the display of the layout while it retrieves the graph behind the scenes. This strategy boosts the user experience by reducing both the true and perceived loading time.

Adopting a similar strategy that prioritizes the loading of that bit of information necessary to render something useful on the screen and, thus, engage the user, can prove particularly useful to any visualization application that requires excessive loading time. People can define a duration only when there is a clear start time and a clear end time (Seow, 2008). As a result, when users get to a point where they finally see something rendered on the screen that they can interact with, they naturally and mentally assume as the end. The rest of the loading can continue behind the scenes.

While browsing pathways, users often go back and forth among several pathways of interest, causing the viewer to load and show the same diagram several times in a relatively short period of time.

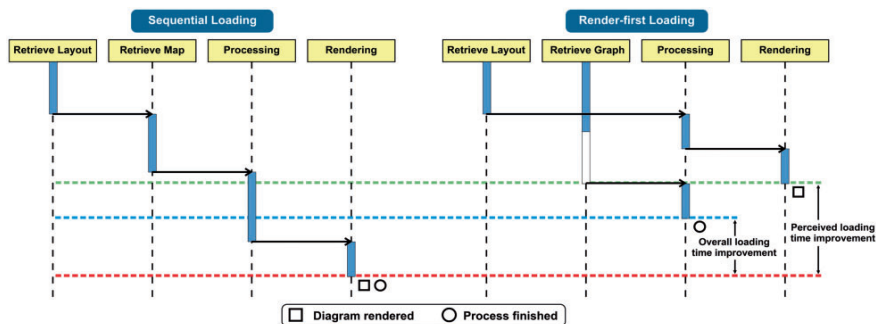


Fig. 2. UML sequence diagram comparing sequential and render-first loading strategies. The difference between the blue and red lines shows the true loading time improvement. The improvement in the perceived loading time is highlighted by the difference between the green and red lines

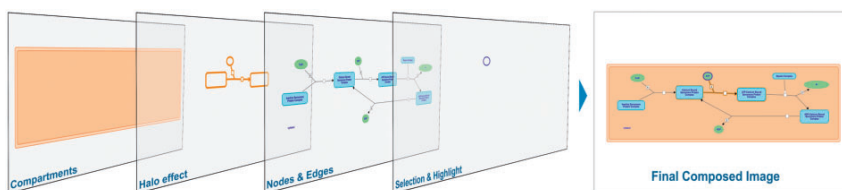


Fig. 3. A simplified example of the adopted multi-layer canvas strategy. First four images from left to right represent different layers composing the final image: (1) Cellular compartments, (2) Halo effect, (3) Nodes & Edges and (4) Selection and Highlight. The rightmost image shows the pathway diagram as seen from the user's perspective

A pathway diagram that has been loaded is very likely to be revisited shortly after visits to other pathways. In computer science, this is known as locality of reference (Denning, 2005), being a very clear use case for cache mechanisms. Hence, the diagram viewer implements a Least Recently Used (LRU) caching mechanism (Denning, 1968) to keep the layout and the view status (zoom level and panning) of the most recently viewed diagrams. When a diagram is revisited, the viewer does not need to request data from the server but uses the cached one in order to display the content as the user previously left it.

2.4 Multi-layer HTML5 canvas strategy

The new version of the diagram viewer responds to common user actions, such as hovering over an element with the mouse and selecting an entity in the diagram, by highlighting the hovered element and marking the selected entity, respectively. Aiming to provide a richer user experience and visually reinforce user actions, the diagram viewer draws a halo around the elements (reactions and participating entities) related to the selection. In addition, when the user selects an entity that is repeated in the same diagram, the viewer marks all instances of that entity as selected and draws halos around all elements related to them.

To improve the visual feedback and optimize the diagram rendering process, the new version of the viewer implements a set of advanced techniques developed and used by the gaming industry. In particular, the multi-layer canvas approach (www.ibm.com/developerworks/library/wa-canvashtml5layering) was adopted to reduce

the processing and redrawing overhead inherent to a single canvas update. Each of the stacked canvases in Figure 3 represents a conceptual layer and is reserved for drawing specific types of glyphs corresponding to different diagram objects such as compartments, reactions, nodes, entities or interactors. By employing this technique, only layers that require redrawing are updated, resulting in reduced rendering times in actions like highlighting or selection. This contributes to enhancing the user experience due to a more responsive behavior.

For instance, while the user moves the mouse pointer across a diagram, only the 'Selection and Highlighting' layer needs to be updated in order to reflect the changes in the highlighted element. Similarly, in case a diagram element is selected, only the 'Halo effect' and 'Selection and Highlight' layers need to be updated. Other resources featuring interactive visualizations that contain a lot of elements can take advantage of this strategy to improve the user experience.

2.5 Space partitioning data structure

Identifying the elements under the mouse pointer is a computationally demanding task if it is performed by a brute force or exhaustive search algorithm (Knuth, 1997). The cost of an exhaustive search algorithm is a linear function of the number of elements to be searched, $O(n)$ in big O notation. Determining whether the mouse pointer position intersects with the area each element occupies can be slow, delaying the action of highlighting and making the interface appear unresponsive to the user.

1212

A.Fabregat et al.

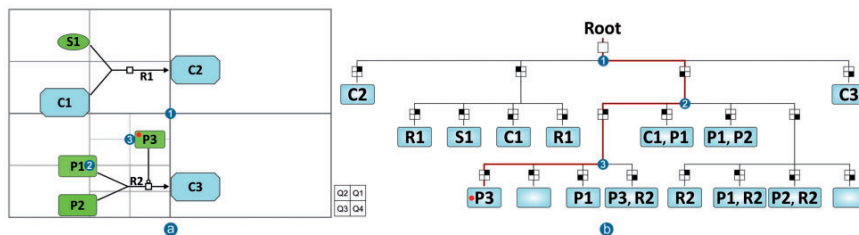


Fig. 4. Hypothetical diagram composed of two separate reactions where (a) shows how the viewport is recursively split into different quadrants, so each of them contains two or less elements, (b) is the representation of the resulting QuadTree to achieve the two-dimensional space partitioning while (c) presents the same diagram elements placed in a normal collection for comparison purposes. The red dot in (a) represents the mouse pointer location and the red path in (b) depicts the tree traversing steps to narrow down the elements to be checked against the mouse location

To speed the search of the hovered element, our new implementation employs a space partitioning data structure, an approach often used to optimize performance. The main advantage of this data structure is that it provides a much less computationally intensive way to query for elements present in a given point or area in space, with a cost that is a logarithmic function of the number of elements to be searched ($O(\log n)$ (Agarwal and Erickson, 1998).

Here, we employed a QuadTree, a tree data structure used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions (Finkel and Bentley, 1974). The QuadTree is employed to efficiently (i) query only those diagram entities present in the viewport that need to be rendered and (ii) identify the entities hovered over, or selected by the mouse without having to follow the brute force method and exhaustively check every diagram object.

Figure 4 provides an example of how the elements in a diagram are located in a QuadTree with quadrant size 2, meaning that only two objects are allowed per quadrant. The red line in Figure 4b highlights the path traversed in the tree to identify the element under the mouse pointer (red dot) based on a series of quick comparisons between the mouse coordinates and every quadrant center starting for the root (center of the viewport) the red dot (Fig. 4a) is the 3rd quadrant (Q3); from the center of Q3 the red dot is in the first quadrant (Q1); from the center of Q1 the red dot is again in its first quadrant (Q1). Since this last quadrant is not further split, the position of the mouse pointer only needs to be compared against the contents of that quadrant, which in this case is only P3. Thus, determining that P3 is the element hovered over by the mouse pointer takes three quadrant comparisons and checking only one element of the nine present in the diagram. This provides a significant improvement over the brute force method that would check the mouse position against every element present in the diagram.

For the new diagram viewer, the QuadTree was extended to work not only with points but also with shapes that occupy diagram areas. The aim was to use it in order to narrow down the number of elements to be drawn depending on whether they are in the part of the diagram visible in the client viewport. This allows a fast, selective redraw limited to visible regions of the diagram, again improving interactivity of the diagram viewer.

Hence, the usage of this data structure could prove particularly useful for other resources featuring interactive visualizations that contain a lot of elements where the requirements include one or more of the following features: (i) determining the element hovered over by the mouse pointer, (ii) determining the selected element

upon user's click or tab action, (iii) smooth animated view transitions or (iv) progressive zoom.

2.6 Renderer delegates

In order to tackle users' requests for less cluttered pathway diagrams, but at the same time preserve access to all information stored in Reactome knowledgebase, the new viewer enables the user to control the flow of visualized information through the level of zoom. This practically means that depending on the zoom level, the viewer enriches or abstracts layers of information. Thus, each diagram entity is rendered in a slightly different way according to the level of zoom, progressively revealing more details as the user zooms in. For instance, as illustrated in Figure 5, common 'house-keeping' molecules, such as ADP, ATP, AMP, water, etc., are hidden in the zoomed out view, resulting in simpler and less crowded diagrams, as explicitly requested by our users.

This strategy also improves rendering time when many elements are in the viewport because fewer details are drawn. Other simplifications are to avoid rounded corners, only showing reaction backbones without central decorators, or removing node attachments or stoichiometry. As users zoom in to specific areas, the number of elements in the viewport falls and more detail is added.

The adoption of this strategy could prove particularly useful for other resources featuring complex visualizations as it allows controlling the granularity of information displayed for a given object and level of zoom. Usability-wise, this enables resources to show different views of the same element based on the zoom, determining the optimal level of detail and type of information to be displayed in each case.

3 Results and discussion

The new pathway diagram viewer combines the set of strategies and data structures described above to improve performance and to include new features that aim to address the shortcomings of the previous version highlighted by the usability testing sessions. The updated diagram storage format combined with the improved 'Render-first' loading strategy resulted in faster loading of diagrams. Additionally, faster rendering was accomplished via the combined use of (i) a QuadTree that efficiently filters down the elements to be drawn based on the visible area, (ii) rendering delegates that declutter the view by regulating the level of detail to be drawn depending on the number of visible elements and (iii) a multi-layer html5 canvas strategy that optimizes rendering by updating only the layers

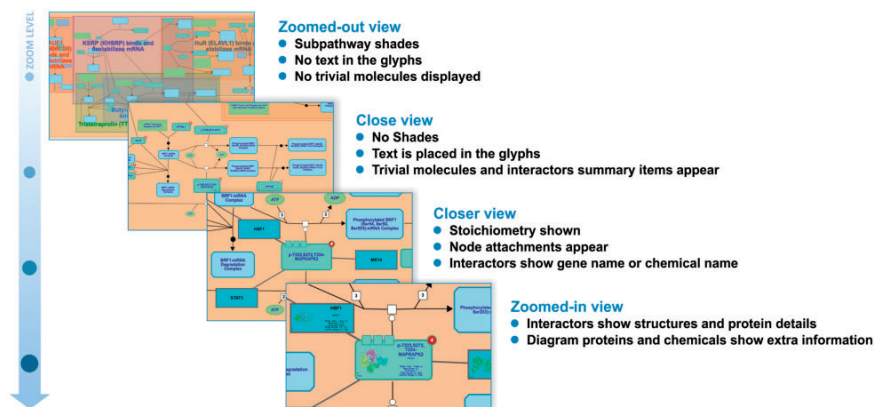


Fig. 5. In the new pathway diagram viewer, the flow of displayed information is controlled through zooming in and out. As a result, depending on the zoom level, the viewer abstracts or enriches the view with layers of information

that require redrawing. Optimized rendering enabled the introduction of animation and smooth transitions that, in turn, help users to maintain diagram context while navigating through pathways. The use of an underlying graph structure provided the basis for improving the built-in search feature, by including all the participating molecules of the pathway whether or not they are visible in the diagram.

Updating the underlying storage format had a positive impact on the performance of the new version of the pathway diagram viewer. To assess this performance boost, we compared the resulting file sizes for both the previous (XML) and the new data format (JSON), as well as the respective times required by the client to process them. This included the time required to populate the model in the client with the diagram data once they were retrieved from the server.

To measure the improvement in performance, a series of experiments were conducted and the results are presented graphically in Figure 6. In particular, Figure 6a presents a chart comparing the file sizes of the Reactome diagrams against the total number of graphical entities present in them for both XML and JSON data format. As expected, for any given pathway diagram, its JSON version has a smaller file size compared to its XML version.

Figure 6b presents a comparison between the times required by the previous (2) and current (3) versions of the client to process diagrams stored in XML and JSON format, respectively, against the number of the diagram entities. The new client requires significantly less time to process any given diagram, which can be attributed to JSON's smaller file size as well as its natural mapping to JavaScript objects, which eliminates the need for complex parsing infrastructure.

The update in the storage format combined with the new render-first loading strategy contributed to reducing the overall diagram loading time, as it is perceived by the user. This includes the time required until the diagram is loaded and fully rendered by the client. Figure 6c presents a chart comparing the times required by the previous and the new version of the client to display diagrams stored in XML and JSON format respectively against the diagram size (measured in number of entities present in a diagram).

A striking feature of the comparison of perceived loading times (Fig. 6c) is that the new diagram viewer is both faster and more consistent. One can easily notice that the times measured for the

previous diagram viewer exhibit high variability, especially for smaller diagrams. This can be explained by the fact that the number of items to be drawn in diagram does not represent the actual size of the pathway in terms of participating molecules. Complexes and sets often contain several participating molecules, and encapsulated pathways might also contain a large number of participants. Also, taking into account that the top-level pathways, in the Reactome event hierarchy, are represented with diagrams which mostly contain subpathways, it is expected that they will contain a quite large number of participating molecules. This fact combined with the previous sequential loading strategy, presented in Figure 2, provides a simple explanation for those relatively small pathways, with only a few entities, that require up to 1.5 s to load. Simply put, before rendering anything on screen, the previous client had to retrieve and parse a large amount of information in order to create the map of all participating entities.

As illustrated in Figure 6c, the new version of the pathway diagram viewer achieves better performance in any given Reactome diagram. In particular, the new version of the client accomplishes loading and rendering of 97% of the diagrams in Reactome in less than 1 s (versus 57% previously); 74% of the total number of diagrams are loaded and rendered in under 0.5 s (versus 31% previously). As previously stated, keeping the application's response times as low as possible has a positive impact on the user experience. This is particularly the case in a web application that is supposed to run inside a web browser environment, where most of its code is executed in a single thread, without use of concurrency. As a result, the adoption of the multi-layer html5 canvas strategy and the space partitioning data structure contributed to minimize CPU workload and therefore allowed room for new features such as animated transitions to be included without penalising the user's experience.

We have conducted a usability testing session centered on the improvements described here (Supplementary Table S1). Users appreciated the animated transitions and progressive zoom functionality as they allowed for smoother and easier navigation. Users also found the new diagram viewer more responsive as it reacted to common user actions by highlighting a hovered element and marking a selected entity. Users did not express concern about crowded/complex

1214

A.Fabregat et al.

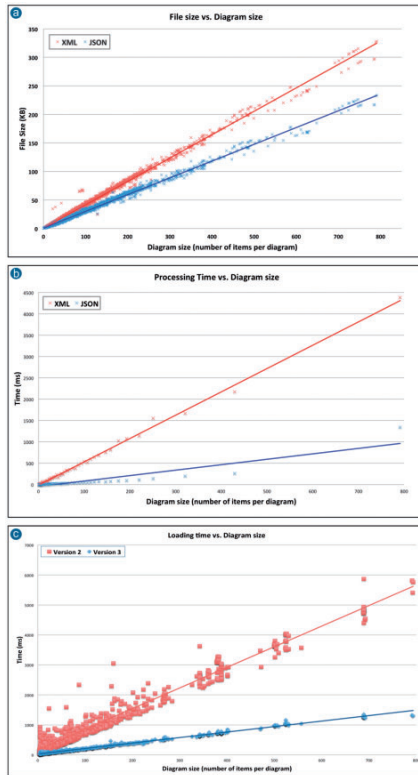


Fig. 6. (a) Comparison of file sizes for both XML and JSON formats versus the diagram size in terms of number of entities present in a diagram. (b) Comparison of processing times achieved by Diagram Viewer v2.0 (consuming diagrams in XML) and Diagram Viewer v3.0 (consuming diagrams in JSON) versus the total number of diagram entities. (c) Comparison of perceived loading times achieved by Diagram Viewer version 2 (consuming diagrams in XML) and version 3 (consuming diagrams in JSON) versus the diagram size (in number of diagram entities). Measured over all human pathway diagrams from Reactome data version 52

pathway diagrams, but did react positively to the features in the new diagram size that enable users to control the amount of detail displayed by zooming in and out. Regarding our improved search functionality, users made positive comments on the fact that they could now search for entities that were indirectly part of a given diagram such as members of a complex or set.

The new diagram viewer was developed in a way that can be both extended or easily integrated as it is in third party applications. Currently, Reactome offers two options for integrating this pathway diagram viewer in other web applications; either using the GWT implementation or the JavaScript wrapper. More details and examples on how to reuse this as a widget can be found at <https://reactome.org/dev/diagram/>.

Conclusions

Through the use of highly optimized data structures and algorithms, Reactome has improved the pathway diagram viewer in terms of performance and usability. The new version of the diagram viewer provides a robust, scalable solution to pathway visualization that is easily integrated into third party applications.

Funding

National Institutes of Health BD2K grant (U54 GM114833); National Human Genome Research Institute at the National Institutes of Health (U41 HG003751); European Bioinformatics Institute (EMBL-EBI); Open Targets (The target validation platform); Medicine by Design (University of Toronto). Funding for open access charge: National Institutes of Health (U54 GM114833). The funding bodies had no role in the design or conclusions of the study.

Conflict of Interest: none declared.

References

- Agarwal,P.K. and Erickson,J. (1998) Geometric range searching and its relatives. *Adv. Discret. Comput. Geom. Am. Math. Soc.*, **23**, 1–56.
- Boci,L. et al. (2012) Comparison between JSON and XML in applications based on AJAX. In: *International Conference On Computer Science and Service System*, Nanjing, China.
- Cerami,E.G. et al. (2011) Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Res.*, **39**, D685–D690.
- D'eustachio,P. (2011) Reactome knowledgebase of human biological pathways and processes. *Methods Mol. Biol.*, **69**, 49–61.
- Denning,P.J. (1968) The working set model for program behavior. *Commun. ACM*, **11**, 323–333.
- Denning,P.J. (2005) The locality principle. *Commun. ACM*, **48**, 19–24.
- Fabregat,A. et al. (2016) The reactome pathway knowledgebase. *Nucleic Acids Res.*, **44**, D481–D487.
- Fabregat,A. et al. (2017) Reactome pathway analysis: a high-performance in-memory approach. *BMC Bioinformatics*, **18**, 142.
- Finkel,F. and Bentley,J.L. (1974) Quad trees: a data structure for retrieval on composite keys. *Acta Inf.*, **4**, 1–9.
- Gawron,P. et al. (2016) MINERVA – a platform for visualisation and curation of molecular interaction networks. *Syst. Biol. Appl.*, **2**, doi: 10.1038/npsjbs.2016.20.
- Kanehisa,M. et al. (2014) Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic Acids Res.*, **42**, D199–D205.
- Knuth,D. (1997) *The Art of Computer Programming. 3: Sorting and Searching*. 3rd edn. Addison-Wesley, Reading, MA, pp. 396–408.
- Krueger,C.W. (1992) Software reuse. *ACM Comput. Surv.*, **24**, 131–183.
- Kuperstein,I. et al. (2013) NavCell: a web-based environment for navigation, curation and maintenance of large molecular interaction maps. *BMC Syst. Biol.*, **7**, 100.
- Kutmon,M. et al. (2016) WikiPathways: capturing the full diversity of pathway knowledge. *Nucleic Acids Res.*, **44**, D488–D494.
- Le Novère,N. et al. (2009) The systems biology graphical notation. *Nat. Biotechnol.*, **27**, 735–741.
- Nurseitov,N. et al. (2009) Comparison of JSON and XML data interchange formats: a case study. *CAINE*, **9**, 157–162.
- Roto,V. et al. (2009) User experience evaluation methods in academic and industrial contexts. In: *Proceedings of Workshop on User Experience Evaluation Methods Interact'09*.
- Seow,S. (2008) *Designing and Engineering Time: The Psychology of Time Perception in Software*. Addison-Wesley Professional, Boston.
- Wang,G. (2011) Improving data transmission in web applications via the translation between XML and JSON. In: *Third International Conference On Communications and Mobile Computing (CMC)*, April 2011, pp. 182–185.

P.3. REACTOME ENHANCED PATHWAY VISUALISATION

Bioinformatics, 33(21), 2017, 3461–3467
doi: 10.1093/bioinformatics/btx441
Advance Access Publication Date: 6 July 2017
Original Paper

OXFORD

Databases and ontologies

Reactome enhanced pathway visualization

Konstantinos Sidiropoulos¹, Guilherme Viteri¹, Cristoffer Sevilla¹, Steve Jupe¹, Marissa Webber², Marija Orlic-Milacic², Bijay Jassal², Bruce May², Veronica Shamovsky³, Corina Duenas¹, Karen Rothfels², Lisa Matthews³, Heeyeon Song², Lincoln Stein^{2,4}, Robin Haw², Peter D'Eustachio³, Peipei Ping⁵, Henning Hermjakob^{1,6,*} and Antonio Fabregat^{1,7,*}

¹European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton CB10 1SD, UK, ²Ontario Institute for Cancer Research, Toronto, ON M5G 0A3, Canada, ³NYU Langone Medical Center, New York, NY 10016, USA, ⁴Department of Molecular Genetics, University of Toronto, Toronto, ON M5G 0A3, Canada, ⁵Department of Physiology, Medicine and Bioinformatics, NIH BD2K Center of Excellence, University of California, Los Angeles, CA 90095, USA, ⁶State Key Laboratory of Proteomics, Beijing Proteome Research Center, Beijing Institute of Radiation Medicine, National Center for Protein Sciences - Beijing, Beijing 102206, China and ⁷OpenTargets, Wellcome Genome Campus, Hinxton CB10 1SD, UK

*To whom correspondence should be addressed.

Associate Editor: Janet Kelso

Received on March 22, 2017; revised on June 6, 2017; editorial decision on July 3, 2017; accepted on July 5, 2017

Abstract

Motivation: Reactome is a free, open-source, open-data, curated and peer-reviewed knowledge base of biomolecular pathways. Pathways are arranged in a hierarchical structure that largely corresponds to the GO biological process hierarchy, allowing the user to navigate from high level concepts like immune system to detailed pathway diagrams showing biomolecular events like membrane transport or phosphorylation. Here, we present new developments in the Reactome visualization system that facilitate navigation through the pathway hierarchy and enable efficient reuse of Reactome visualizations for users' own research presentations and publications.

Results: For the higher levels of the hierarchy, Reactome now provides scalable, interactive textbook-style diagrams in SVG format, which are also freely downloadable and editable. Repeated diagram elements like 'mitochondrion' or 'receptor' are available as a library of graphic elements. Detailed lower-level diagrams are now downloadable in editable PPTX format as sets of interconnected objects.

Availability and implementation: <http://reactome.org>

Contact: fabregat@ebi.ac.uk or hhe@ebi.ac.uk

1 Introduction

Pathway databases like Reactome systematically associate proteins with their functions and link them into networks that describe the reaction space of an organism. The basic unit of a pathway database is a reaction in which molecules are transformed. Transformations include the chemical changes of intermediary metabolism, as well as chemical modifications of proteins and other macromolecules,

formation and reorganization of complexes, and transport events that move molecules from one cellular location to another. Pathways that accomplish more complex tasks like glycolysis or signal transduction mediated by a tyrosine kinase receptor, can be assembled from these reactions and can be grouped further to describe domains of biology like metabolism or signaling. Functional linkages between processes are consistently visible as shared

molecules and dependencies, e.g. the output of one reaction is the input of another or positively regulates it, within or between pathways and domains.

Pathways in Reactome are organized hierarchically, grouping related detailed pathways (e.g. translation, protein folding and post-translational modification) into larger domains of biological function like metabolism of proteins. This hierarchical organization largely follows that of the Gene Ontology (GO) biological process hierarchy (Ashburner *et al.*, 2000; The Gene Ontology Consortium, 2015). Reactome pathways can be distinguished into two different types: higher level pathways (HLPs) that aggregate pathways within a similar biological process and detailed lower level pathways (LLPs) where the molecular processes are annotated as series of biomolecular reactions. Previously, HLP diagrams (HLDs) were simply implemented as a series of boxes symbolizing the subpathways, and allowing to navigate to them.

A formal data model such as the one embodied in Reactome makes pathways computationally accessible. Additionally, Reactome offers a pathway analysis service that supports enrichment and expression analysis (Fabregat *et al.*, 2016). Using the analysis service users can easily ask, e.g. whether the set of proteins up-regulated in a model system in response to a stress is distributed at random over reaction space or is clustered in particular domains, or could take advantage of the disease visualization to find out whether a mutation that blocks the function of a protein of interest would directly perturb any of the reactions that make up a process of interest. Additional tools are needed, however, to enable biologists to browse database content to visualize the relationships between parts of a domain or to explore possible relationships between domains.

Well-designed pathway visualization tools need to support diverse views providing different levels of detail. Dynamic navigation between views enables human users to visualize connections between pathways and domains (Suderman and Hallett, 2007). Diagrams can support inference more efficiently than equivalent linguistic representations (Perini, 2013). To facilitate interoperability it is also valuable for visualization tools to conform as much as possible to community standards like Systems Biology Graphical Notation (SBGN) (Le Novère *et al.*, 2009).

The challenge of pathway data visualization has been approached by several resources. In cases like MINERVA (Gawron *et al.*, 2016) and NAVICELL (Kuperstein *et al.*, 2013) the Google map engine was adopted to visualize pathways using SBGN. WikiPathways (Kutmon *et al.*, 2016) and KEGG (Kanehisa *et al.*, 2014) display pathways using an in-house developed viewer. Finally, other resources like Pathway Commons (Cerami *et al.*, 2011) display pathways as networks of gene-gene interactions. The most common navigation features used to explore pathway diagrams include zooming, panning and selection of pathway elements to view detailed information. Some tools, such as MINERVA or WikiPathways, also allow users to map drug targets or overlay experimental data. Another popular tool for pathway analysis and visualization is Ingenuity Pathway Analysis (<https://www.qiagenbioinformatics.com>), but as a commercial tool, it is inaccessible to many users.

Reactome's approach to scalable visualization environment, in place since 2015 (Fabregat *et al.*, 2016), provides multiple levels of detail as shown in Figure 1. A user starts with the 'pathways overview' view of all of reaction space, chooses to view the domain 'haemostasis', and within that domain chooses the pathway, 'platelet adhesion to exposed collagen'. The first view is a graph of the entire Reactome event hierarchy (Fig. 1a); the final view shows the molecular details of a reaction sequence in a familiar SBGN-

compliant format close to that of a classic metabolic map (Fig. 1d). The intermediate view, showing the pathways comprising haemostasis as a set of labeled green boxes (Fig. 1b), provides functionality for navigation and data analysis, but conceals the relevant biology, that this process is happening in a damaged blood vessel, that its parts occur in a causal sequence and that they involve complex interactions among molecules and cells in the blood and components of the vessel wall.

In addition to improved navigation, researchers have frequently requested options to export and save their pathways of interest in a format that can be reused for presentations, papers or other purposes, allowing them to easily manipulate the available pathway visualizations (e.g. alter the layout of a diagram or overlay the results of their research).

Here, we present the recent updates in the Reactome web interface providing improved visualization and navigation of Reactome pathways, as well as new options for downloading and re-using the pathway diagrams.

2 Implementation

Aiming to address these challenges and boost the overall user experience, three new features were developed and integrated in the Reactome pathway diagram viewer (Fig. 2): (i) textbook-style enhanced high level diagrams (EHLs), (ii) a mechanism to highlight different subpathways using coloured boxes in zoomed-out views of classic LLP diagrams and (iii) an option to export regular diagrams to PowerPoint.

Figure 2 presents the use of EHLs to support a user who wants to navigate through the Cell Cycle domain of biology to arrive finally at the molecular details of pathway of interest, Mitotic Prophase. This top-down navigation of the Reactome pathways visualization begins with the Pathways Overview which provides a genome wide view of all the pathways in Reactome and their parent-child relationships. As the user progressively zooms in on Cell cycle, a series of EHLs are displayed as interactive textbook-like illustrations that provide a visual representation of key biological concepts.

EHLs aim to assist users to easily identify and focus on areas of interest by taking advantage of the human ability to easily perceive complex but visually obvious concepts. In this example, two levels of HLPs (Mitotic Cell Cycle and M Phase) are traversed through EHLs, leading to the classic pathway diagram of the Mitotic prophase. This diagram, initially presented in its zoomed-out view, includes coloured boxes that highlight its different subpathways, helping the user to easily identify and comprehend the internal parts of this biological process. Subpathway highlighting progressively fades out as the user further zooms into the diagram and focuses on the reactions of a specific subpathway, for example Cisternae Pericentriolar Stack Reorganization.

2.1 Implementation of interactive EHLs

Reactome previously used very simple sets of green box icons to facilitate navigation from highly abstract to more concrete, detailed levels of the pathway hierarchy (Fig. 1b). In some cases these were supplemented by static illustrations depicting the relationship of these pathways to each other (Fig. 1c). These static illustrations are often visually striking images of biological processes, but they lack interactivity; users cannot navigate to contained subpathways nor overlay summarized views of user data.

To enhance the HLP illustrations and make them interactive in the scope of the Reactome pathway browser, a number of technical

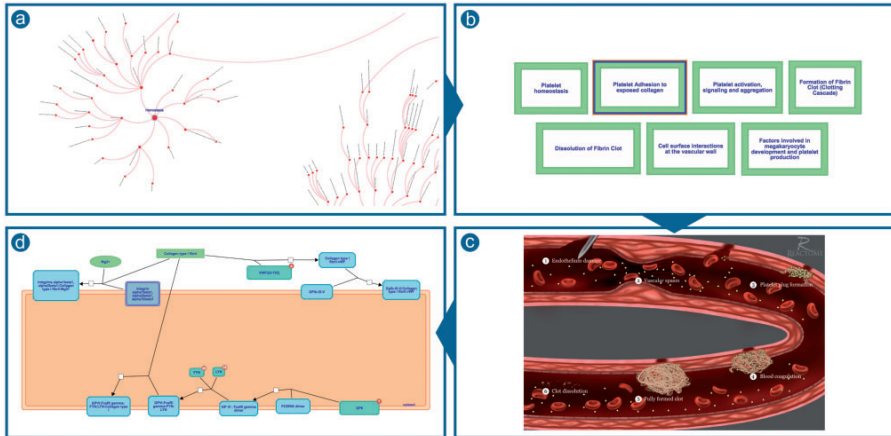


Fig. 1. Reactome's previous approach to scalable visualization (a) Pathways overview (b) HLDs were shown as a set of green boxes (c) Static images of biological processes illustrations were provided for some of the diagrams (d) SBN-compliant diagrams containing molecular details of a reaction sequence

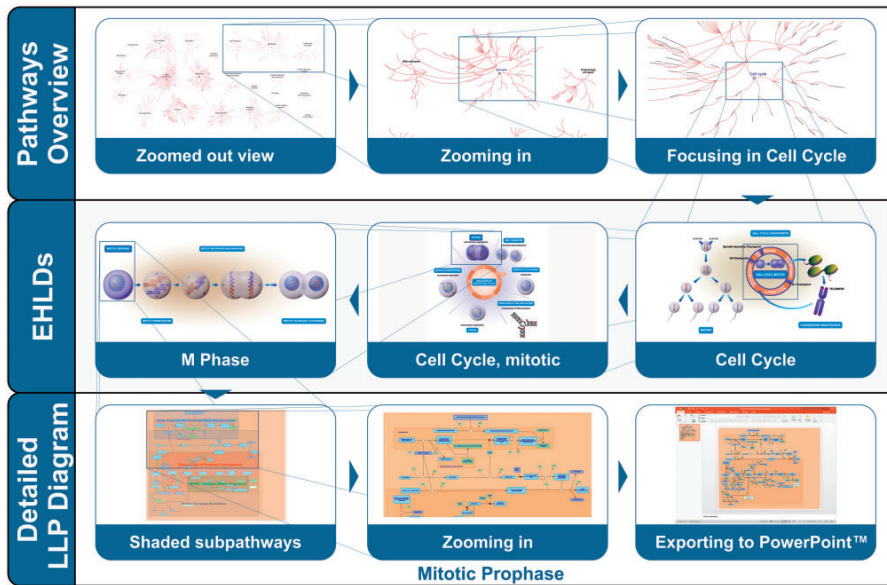


Fig. 2. Different levels of visualization address different scenarios. The top row shows navigation from the entirety of reaction space to focus on Cell Cycle. The middle row presents EHLs displaying successively narrower aspects of the cell cycle process. The bottom row presents navigation down to molecular details of individual reactions involved in mitotic prophase, and the optional export of this view to Microsoft PowerPoint

and conceptual requirements had to be considered. Technically, the illustration had to be stored in a format that could be easily processed, while conceptually the illustration needed to provide a one-to-one mapping between the annotated pathway hierarchy and the graphical elements.

The task of producing these new EHLs was assigned to a team of expert curators and illustrators who worked closely together to generate high quality overview diagrams for the higher levels of the Reactome pathway hierarchy. A common style and iconography was adopted wherever possible to enhance the 'recognition effect' and facilitate efficient navigation.

On the technical side, the SVG format was selected mainly due to the advantages it presents over other graphic formats such as PNG or JPEG. These advantages include (i) object-based vector representation for easy editing, (ii) resolution-independent zooming (Battiatto et al., 2005) suitable for the observation of large integrated pathways, (iii) interaction features for richer interfaces, (iv) data in regular text format as a subset of an eXtensible Markup Language (XML) that can be easily handled by computer programs or text editors and (v) support by most popular web browsers. Additionally, most popular graphic software packages used by designers, such as Adobe Illustrator, Corel Draw, Inkscape, can be used to import, edit and export in SVG format.

In the detailed LLP diagrams, the large number of entities and the variety of custom overlaid features made the usage of a multi-layered HTML5 Canvas an appropriate technology (Miller et al., 2013). However, since EHLs have a limited number of entities, simple overlay features and a simpler rendering strategy, where the flow of information does not depend on the level of zoom, the adoption of an SVG renderer was deemed to be the best solution because modern browsers directly render this format. This allowed developers to focus on features such as overlays, zoom or translation which were implemented by applying a series of filters and transformations.

As part of the Reactome web interface, we have implemented an SVG rendering component that, beyond standard zooming and panning, allows specific regions of the diagram to be highlighted, recognizes mouse click events on specific regions to allow navigation to subpathways and allows analysis results to be overlaid onto the diagram. For the purposes of EHLs, all of the required technical annotations were included in the SVG files by using the database id attribute of each relevant diagram element. In particular, three different types of technical annotations were used: (i) regions that represent a specific subpathway, (ii) labels containing the name of that subpathway and (iii) the region to be overlaid with the analysis results. Any element with no technical annotation was considered a decoration. The EHL viewer reads the SVG file, renders the content and based on the technical annotations sets up the active regions, which can later be highlighted/selected or overlaid with analysis results. By interacting with any of the active regions representing subpathways, users can navigate to the respective subpathway diagram. The relevant code is part of the EHL package available on the Reactome public GitHub repository (<https://github.com/reactome-pwp/diagram>). In addition, EHLs are also available in the reusable stand-alone JavaScript diagram viewer (<http://reactome.org/dev/diagram/js>).

In the example EHL presented in Figure 3b, the Haemostasis pathway hierarchy is no longer represented as a set of green boxes. Instead, the EHL combines the illustration logic with a new design which includes a one-to-one mapping of its subpathways. This EHL is processed by the software and placed as a fully interactive diagram skin, providing all the features of regular pathway diagrams

such as hovering over elements, selection, flagging and pathway analysis results overlay. Moreover, the user interface provides fast zooming and panning support, with no image quality degradation. To provide easier in-diagram navigation, the viewer features a small thumbnail at the bottom-left corner of the viewport. EHLs build on the power of illustrations to convey the depicted biological processes and their causal relationships, by adding interactivity and providing a rich user experience. For instance, the EHL of Haemostasis (Fig. 3b) provides a clearer visual description regarding the role and order of each of the seven subpathways.

The viewer allows users to overlay pathway analysis results onto EHLs (Fig. 3c). The results are displayed in the label of each subpathway. The label is overlaid by a coloured rectangular shape; its width and colour represent the percentage of hit entities and the P -value, respectively (Fig. 3d). Subpathways with a P -value below a certain threshold ($P < 0.05$) are coloured in grey. For hit subpathways, additional information about the hit elements and the false discovery rate are displayed next to the label. Upon hovering or selection of a subpathway, its P -value is indicated in the coloured legend bar displayed on the right side of the viewport. Analysis results are temporarily stored on the Reactome server. The storage period depends on usage of the service but is at least 7 days. Stored results are available via the token assigned to the results file when it is created and displayed in the URL for the results report. Users can easily share their view of Reactome with the results of their analysis overlaid by simply sharing the URL (Fabregat et al., 2016).

EHLs, including any analysis result overlay present at the time, can be easily downloaded and saved in SVG format. Users can edit the content of the exported files through the use of commercial or open-source graphics applications. EHLs use a consistent iconography that reuses glyphs when the same entity plays a role in more than one biological process. For example, all platelets in the Haemostasis EHL are represented by the same symbol. We have made a library of these graphical elements to provide them in SVG, PNG and EMF formats. The icon library is available at <http://reactome.org/icon-lib> and is distributed under the terms of the CC-BY license (<https://creativecommons.org/licenses/by/4.0/>). The aims of providing such a library are to facilitate the creation of uniform diagrams through the use of pre-existing glyphs and to offer these components to the community for reuse. Figure 4 presents a small sample of the elements that are available through the Reactome iconography library.

Researchers can use the provided icons to create their own illustrations to convey their findings and ideas, whether in a pathway diagram or a grant application illustration. Additionally, we would also like to encourage the community to contribute to the extension of this library through new glyphs (<http://reactome.org/icon-info>).

2.2 Subpathway highlighting

Coloured boxes that highlight and distinguish subpathways within an LLP have been implemented to support easier navigation. Figure 5 compares the same pathway before (a) and after (b) the inclusion of this functionality. The boxes highlight the specific diagram reactions belonging to each subpathway, allowing further zooming in to the areas of interest within a given pathway diagram.

The position and colour of boxes is calculated on the server side at database release time and persists for the 3-month lifetime of the release. This strategy supports fast loading and ensures that the look and feel remains the same. The most complicated part of the task of automatic subpathway highlighting is deciding where to place the text inside each box. The adopted algorithm follows a basic space

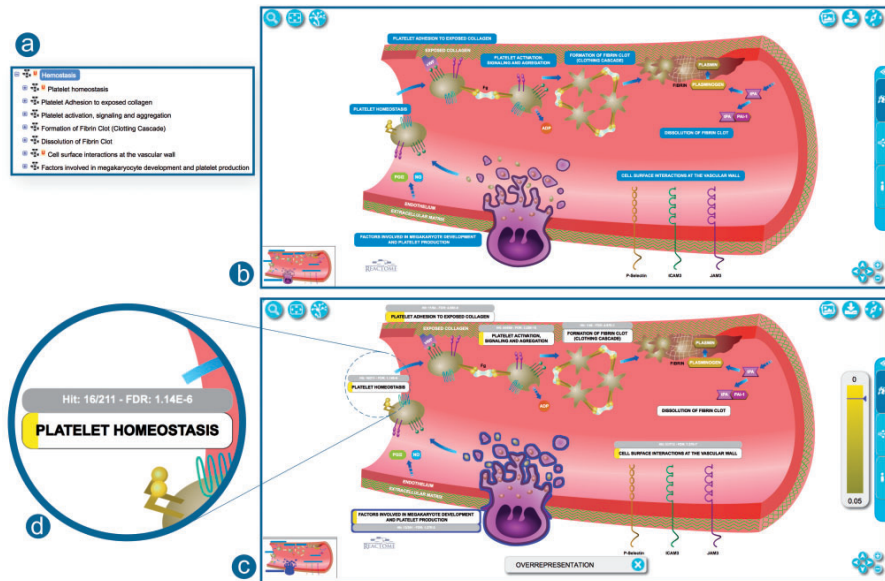


Fig. 3. Correspondence between the hierarchy and EHL. (a) Pathway hierarchy view for Haemostasis. (b) Haemostasis as an EHL representation. (c) Haemostasis EHL overlaid with pathway enrichment analysis results. The width of the yellow bar under the pathway label indicates the proportion of pathway entities contained in the analysed dataset. (d) A closer view to 'Platelet Homeostasis' label with analysis results overlaid. <http://reactome.org/PathwayBrowser/#/R-HSA-109582>

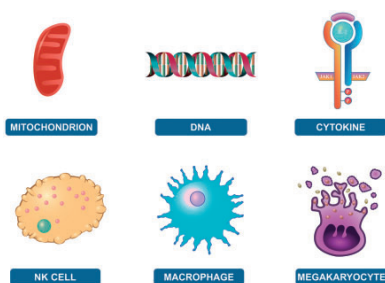


Fig. 4. Example of different elements provided in the iconography library. <http://reactome.org/icon-lib>

partitioning approach where the text is placed in the largest and widest possible rectangle inside each subpathway box after calculating the overlap with all the other subpathway boxes.

2.3 Export options

In addition to easy visualization and navigation, users often want to export/save pathways of interest in a format that can be reused for presentations, publications, or other purposes. In some cases, users

want to use the pathway layout without modification, but in other cases, they may want to alter the diagram content to show the results of their own research or alter the layout. Exporting static images such as PNG files covers the first use case but to cover the second requirement, other formats are needed.

When exporting to other editable formats, there are several options to consider. A UX testing session with domain experts conceived the idea of exporting diagrams as an interconnected set of objects that adapt to layout changes. The main requirement was that when a glyph is moved around all the connected objects must follow e.g. when the user moves the glyph of a given reaction product, the reaction output line has to automatically readjust to keep pointing to it. Office tools such as Microsoft PowerPoint (<http://office.microsoft.com/PowerPoint>) are very common among biologists and allow the creation of multimedia presentations where the user can rearrange the contained objects and customize other properties such as size, colour or shape. By exporting Reactome diagrams to these tools in a standard format (PPTX), users can take advantage of all these features. Other options as SVG or PDF would export the content of a given diagram in a set of non-interconnected objects making changes in layout more difficult due to the need of manually moving every affected object.

Several issues were addressed to enable PPTX export. Reactome diagrams follow the SBN standard, but the internal Reactome diagram data model does not define interconnections between different objects or anchor points between reactions, shapes and their participants. This absence of interconnection between elements needed to

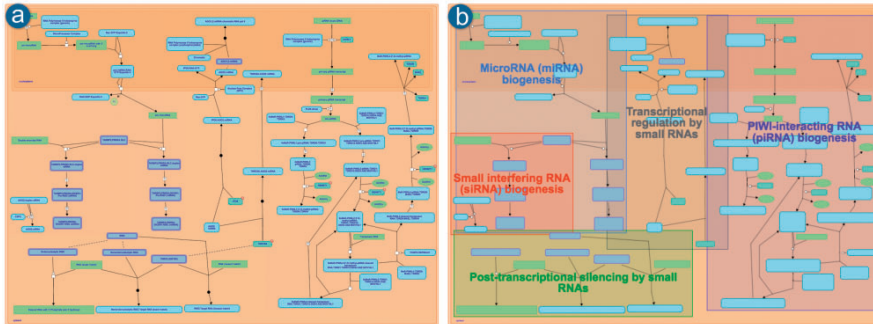


Fig. 5. Pathway diagram for Gene silencing by RNA. (a) As it was displayed prior to the inclusion of the subpathways highlighting. (b) After adding the feature of highlighting the existing subpathways in the zoomed-out view (where the text in the entity icons is also omitted). <http://reactome.org/PathwayBrowser/#/R-HSA-211000>

be programmatically addressed during the conversion phase to achieve movability of the contained objects without affecting their relationship with connecting lines. Different techniques were employed, including the use of invisible anchor points to join the segments of a reaction object and group the objects that visually define reaction properties (i.e. catalysis or regulation line endings).

The PPTX files are generated on the server side, using the colour profile selected by the user on the client side. Storing the content in PPTX format is not straightforward, so Reactome utilized Aspose.Slides (<https://www.aspose.com/products/slides/java>), a commercial JAVA API for reading, writing and manipulating PowerPoint documents. Figure 6 presents an example of a regular diagram exported to Microsoft PowerPoint.

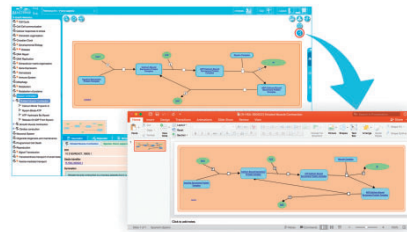


Fig. 6. The pathway diagram for Striated Muscle Contraction exported to Microsoft PowerPoint. (<http://reactome.org/PathwayBrowser/#/R-HSA-390522>)

3 Discussion

The new features presented above have been included in the Reactome Pathway Diagram Viewer, which is fully integrated in the current version of the Reactome Pathway Portal. EHLs and the option to export diagrams to PowerPoint are available to users accessing Reactome through the Pathway Portal or through the standalone version of the Diagram Viewer widget, which is available for integration in third party applications (<http://reactome.org/dev/diagram>).

Reactome curators and designers have created informative and visually appealing EHL illustrations. Feedback from users regarding this new feature has been very positive. In particular, users found the new, interactive graphical representation of Reactome pathways more descriptive of the biological process and consequently more intuitive to navigate when compared to the previous static HLDs.

EHLs have been designed to contain images that will be familiar to biologists. They use a consistent iconography that is based on a survey of typical textbook and online representations of the process. The intention is to make the navigation experience more intuitive and visually pleasing; the user will recognize the process that is represented and be able to select the appropriate region of the illustration to navigate to the next level of the Reactome hierarchy level without the need to read and understand text labels; ultimately the user will arrive at a classic, detailed pathway diagram that represents the molecular mechanism underlying the pathway. For users

who are not familiar with the graphical representation used in EHLs the text labels are retained.

Reactome users have responded positively to the inclusion of the subpathways highlight feature. Specifically, users found the new method of rendering LLDs (Fig. 5) more descriptive and less cluttered. The coloured boxes around subpathways give users a bird's eye view of the displayed pathway, and to identify and later focus on regions of interest.

Apart from requesting enhancements of web visualization, users have often reported that the ability to export to PowerPoint would be a useful feature, mainly because it would enable them to conveniently open and edit Reactome pathway diagrams with the tool they use for creating posters and presentations. Feedback was particularly positive in relation to the ability to reposition diagram entities.

The limited visual appeal and navigability issues of previous Reactome HLDs have been addressed by the introduction of EHLs, which represent biological processes in a familiar textbook style that allows intuitive navigation to more specific sub-topics. In addition, EHLs are used to represent summarized analysis results. We have provided the ability to export EHL images and associated analysis result overlays in a lossless, editable format, enabling users to represent their own research results in the context of Reactome pathway diagrams. In addition, the Reactome pathway iconography library provides graphical representations of common molecular

biology elements suitable for use in slides and publications. Finally, classic pathway diagrams can now be exported in PPTX format, allowing their editing and reuse with familiar presentation software.

Regarding future work, in the short term Reactome curators and designers will replace all HLDs (ca. 85) with textbook-style EHLs. In the mid-term, we plan to improve the integrated diagram search feature by taking advantage of Solr and our graph database (<http://reactome.org/dev/graph-database>). Finally, in a more generic fashion, we will continue improving the user interface to make it more user-friendly and responsive to the user's behaviour and environment such as screen size, platform and orientation.

Funding

National Institutes of Health BD2K grant (U54 GM114833); National Human Genome Research Institute at the National Institutes of Health (U41 HG003751); European Bioinformatics Institute (EMBL-EBI); Open Targets (The target validation platform); Medicine by Design (University of Toronto). Funding for open access charge: National Institutes of Health (U54 GM114833).

Conflict of Interest: none declared.

References

Ashburner, M. *et al.* (2000) Gene ontology: tool for the unification of biology. *The Gene Ontology Consortium. Nat. Genet.*, **25**, 25–29.

- Battiato, S. *et al.* (2005) SVG rendering for Internet imaging. In: *Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception, CAMP'05, July 04–06, 2005*, Palermo, Italy, pp. 333–338.
- Cerami, E. G. *et al.* (2011) Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Res.*, **39**, D685–D690.
- Fabregat, A. *et al.* (2016) The Reactome pathway Knowledgebase. *Nucleic Acids Res.*, **44**, D481–D487.
- Gawron, P. *et al.* (2016) MINERVA – a platform for visualisation and curation of molecular interaction networks. *Syst. Biol. Appl.*, **2**, 16020.
- Kanehisa, M. *et al.* (2014) Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic Acids Res.*, **42**, D199–D205.
- Kuperstein, J. *et al.* (2013) NaviCell: a web-based environment for navigation, curation and maintenance of large molecular interaction maps. *BMC Syst. Biol.*, **7**, 100.
- Kutmon, M. *et al.* (2016) WikiPathways: capturing the full diversity of pathway knowledge. *Nucl. Acids Res.*, **44**, D488–D494.
- Le Novère, N. *et al.* (2009) The Systems Biology Graphical Notation. *Nat. Biotechnol.*, **27**, 735–741.
- Miller, C. A. *et al.* (2013) Scrib: an HTML5 Canvas-based graphics library for visualizing genomic data over the web. *Bioinformatics*, **29**, 381–383.
- Perini, L. (2013) Diagrams in biology. *Knoul. Eng. Rev.*, **28**, 273–286.
- Suderman, M., and Hallett, M. (2007) Tools for visually exploring biological networks. *Bioinformatics*, **23**, 2651–2659.
- The Gene Ontology Consortium (2015) The Gene Ontology Consortium Gene Ontology Consortium: going forward. *Nucleic Acids Res.*, **43**, D1049–D1056.

P.4. REACTOME GRAPH DATABASE: EFFICIENT ACCESS TO COMPLEX PATHWAY DATA

RESEARCH ARTICLE

Reactome graph database: Efficient access to complex pathway data

Antonio Fabregat^{1,2*}, Florian Korninger¹, Guilherme Viteri¹, Konstantinos Sidiropoulos¹, Pablo Marin-Garcia^{3,4}, Peipei Ping⁵, Guanming Wu⁶, Lincoln Stein^{7,8}, Peter D'Eustachio⁹, Henning Hermjakob^{1,10*}

1 European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, United Kingdom, **2** Open Targets, Wellcome Genome Campus, Hinxton, United Kingdom, **3** Fundación Investigación INCLIVA, Universitat de València, Valencia, Spain, **4** Instituto de Medicina Genómica, Valencia, Spain, **5** NIH BD2K Center of Excellence and Department of Physiology, Medicine and Bioinformatics, University of California, Los Angeles, California, United States of America, **6** Oregon Health and Science University, Portland, Oregon, United States of America, **7** Ontario Institute for Cancer Research, Toronto Canada, **8** Department of Molecular Genetics, University of Toronto, Toronto, Canada, **9** NYU Langone Medical Center, New York, New York, United States of America, **10** State Key Laboratory of Proteomics, Beijing Proteome Research Center, Beijing Institute of Radiation Medicine, National Center for Protein Sciences, Beijing, China

* fabregat@ebi.ac.uk (AF); hhe@ebi.ac.uk (HH)



OPEN ACCESS

Citation: Fabregat A, Korninger F, Viteri G, Sidiropoulos K, Marin-Garcia P, Ping P, et al. (2018) Reactome graph database: Efficient access to complex pathway data. *PLoS Comput Biol* 14(1): e1005968. <https://doi.org/10.1371/journal.pcbi.1005968>

Editor: Timothée Poisot, Université de Montréal, CANADA

Received: July 19, 2017

Accepted: January 10, 2018

Published: January 29, 2018

Copyright: © 2018 Fabregat et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: The development of Reactome is supported by grants from the National Institutes of Health (P41HG003751 and U54GM114833) (<https://nih.gov/>), European Bioinformatics Institute (EMBL-EBI) (<https://ebi.ac.uk/>), Open Targets (The Target Validation Platform) (<https://www.opentargets.org/>), and Medicine by Design (University of Toronto) (<http://mbd.utoronto.ca/>).

Abstract

Reactome is a free, open-source, open-data, curated and peer-reviewed knowledgebase of biomolecular pathways. One of its main priorities is to provide easy and efficient access to its high quality curated data. At present, biological pathway databases typically store their contents in relational databases. This limits access efficiency because there are performance issues associated with queries traversing highly interconnected data. The same data in a graph database can be queried more efficiently. Here we present the rationale behind the adoption of a graph database (Neo4j) as well as the new ContentService (REST API) that provides access to these data. The Neo4j graph database and its query language, Cypher, provide efficient access to the complex Reactome data model, facilitating easy traversal and knowledge discovery. The adoption of this technology greatly improved query efficiency, reducing the average query time by 93%. The web service built on top of the graph database provides programmatic access to Reactome data by object oriented queries, but also supports more complex queries that take advantage of the new underlying graph-based data storage. By adopting graph database technology we are providing a high performance pathway data resource to the community. The Reactome graph database use case shows the power of NoSQL database engines for complex biological data types.

Author summary

To better support genome analysis, modeling, systems biology and education, we now offer our knowledgebase of biomolecular pathways as a graph database. We have developed a tool to migrate the Reactome content from the relational database used in curation

Funding for open access charge: National Institutes of Health [U54GM114833]. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

to a graph database during each quarterly release process. The new graph database has two main advantages; higher performance and simpler ways to perform complex queries. Reactome has already adapted its software infrastructure to benefit from this growing in popularity storage technology, significantly improving query efficiency, by reducing the average query time by 93%. We strongly believe that the successful adoption of a graph database by Reactome demonstrates the positive impact this new technology could potentially have in the field and could provide a practical example for other community projects with similar complex data models to move their storage to a graph database while retaining their data models.

This is a *PLOS Computational Biology* Software paper.

Introduction

Reactome (<https://reactome.org>) is a free, open-source, open-data, curated and peer-reviewed knowledgebase of biomolecular pathways. Reactome annotates processes in a consistent pathway model to create an online resource for researchers as a core reusable pathway dataset for systems biology approaches. Reactome provides infrastructure and intuitive bioinformatics tools for search, visualisation, interpretation and analysis of pathways [1].

Reactome contains a detailed representation of cellular processes, as an ordered network of molecular reactions, interconnecting terms to form a graph of biological knowledge. Like most biomolecular pathway knowledgebases, Reactome has relied on a relational database to store its content. Although widely used among pathway knowledgebases for data management, relational databases are not always the best fit to deal with today's performance requirements and increasing data complexity [2, 3]. Relational databases cope well with modeling and storing complex pathway information, but the final product is very likely to contain many intermediate tables to represent many-to-many relationships. As a result, database queries across a network of highly interconnected pathway data are often difficult to formulate and require a high number of join operations, ultimately resulting in degradation of performance and excessive response times.

The Reactome data model naturally forms a large interconnected network that can be seen as a directed graph, which consists of a set of nodes and a collection of directed edges connecting ordered pairs of nodes [4]. Storing Reactome pathway data in its natural form has multiple benefits. Most significantly, it does not require any transformation of data into a flat or denormalised table format. As a result, data can be persisted as originally designed, reducing the complexity of the database and thus allowing a more straightforward access to the Reactome knowledgebase [3].

Here we describe the motivation behind our adoption of a graph database and show how Reactome benefits from this change in the underlying storage technology to overcome the previously mentioned limitations imposed by relational databases. The main target audiences for this manuscript are bioinformatics developers, who might be inspired to apply a graph database in a similar domain, and bioinformaticians involved in pathway analysis, who might benefit from using our graph database directly. While users of the Reactome web interface take advantage of the described gains in performance, features, and stability, the Reactome web interface is described in detail in [1].

Design and implementation

The Reactome data model

Reactome uses a frame-based knowledge representation [5]. The data model (<https://reactome.org/content/schema>) consists of classes (frames) that describe different concepts like reaction or entity. Classes have attributes (slots) that hold properties of the represented class instances, like names or identifiers. The value types contained in the slots can be primitive (string, numbers, or boolean) or references to other class instances. Therefore, knowledge in Reactome is captured as instances of these classes with their associated attributes.

While implementing its relational database, Reactome opted for a physical design that favoured flexibility over performance. Simply put, the relational database incorporated an increased level of abstraction in its physical design resulting in easier adoption of new concepts but at the same time heavily impacting the complexity and execution time of its queries. However, since the graph database natively stores Reactome content in a graph following its model, this trade-off between flexibility and performance is no longer needed.

The Event and PhysicalEntity (PE) classes hold prominent positions in the Reactome model. Events are the building blocks used in Reactome to represent biological processes and are further subclassed into Pathways and ReactionLikeEvents (RLE). RLEs are single-step molecular transformations. RLE includes Reaction among other types like FailedReaction, Polymerisation, Depolymerisation, and BlackBoxEvent. Examples discussed here all involve transformations of the "Reaction" type but all types are handled in the same way with the same results. Pathways are ordered groups of RLEs that together carry out a biological process. PEs are the participants in these events. PE types include SimpleEntity for chemicals, Entity-WithAccessionedSequence for proteins, Complex for multi-molecular structures and EntitySet for PEs grouped together on the basis of their shared function.

Moving from a relational to a graph database

Persistence of a model, like the one described above, can be achieved with flat files, a relational database, or a non-relational database (e.g. a graph database). The selected underlying storage mechanism determines how data are physically stored and accessed. Consequently, each of these options comes with both advantages and disadvantages in terms of performance and scalability. Until recently, Reactome relied on a relational database (MySQL) for both storing its content during curation and accessing it in its production phase. Among the factors that contributed to this decision were that (1) Protégé (<http://protege.stanford.edu>) was used as the curator tool during Reactome's nascent years with a Perl script processing the Protégé files to store content into a MySQL database, which was modeled according to the Protégé schema, (2) at the time a relational database met Reactome's needs for data integrity and consistency, and (3) relational databases were well established for biological data whereas graph based solutions were hardly used in the field [6, 7].

It was not until recently that graph databases became a popular technology in different areas of computational biology. Henkel et al. proposed the concept of graph databases for storage and retrieval of computational models of biological systems [7]. Summer et al. developed a Cytoscape application that takes advantage of the Neo4j database to perform server-side analysis of large and complex biological networks [8]. In [9] the authors explored the potential of using a graph database to facilitate data management and analysis to provide biological context to disease-related genes and proteins. Toure et al. developed a Java-based framework that transforms biological pathways represented in SBGN format into the Neo4j graph database, enabling more powerful management and querying of complex biological networks [10].

Balaur et al. demonstrated that advanced exploration of highly connected and comprehensive genome-scale metabolic reconstructions can benefit from an integrated graph representation of the model and associated data [11]. Swainston et al. described *biochem4j* that enables complex queries by linking a number of widely used chemical, biochemical and biology resources within a graph database [12].

Reactome has gradually introduced a Neo4j graph database (<https://neo4j.com/>) to store and query its content in the production phase since July 2016 (version 57). Neo4j is an open source, transactional and ACID (Atomicity, Consistency, Isolation, and Durability) compliant graph database [13]. Native graph databases, such as Neo4j, naturally store, manage, analyze, and use data within the context of connections to improve performance and flexibility when handling highly interconnected data compared to that in SQL. Neo4j's greatest advantage and probably its most defining feature is Cypher: a declarative, pattern matching query language, specifically designed for dealing with graph data structures [14, 15].

The Reactome knowledgebase has many use cases, like the one in Fig 1, where the use of a graph model together with a query language like Cypher can greatly improve response times and simplify the code necessary to access the data. For instance, recursively retrieving all reactions of a pathway, retrieving the participants of a reaction or a pathway, deconstructing a complex or a set into its participating molecules, or enumerating the chain of consecutive reactions that lead to the formation of a signalling complex are typical use cases that benefit greatly from traversing the graph version of the Reactome knowledgebase.

Fig 1 provides a simplified example where reactions only contain lists of reactants and products, instances of the PE class. In the relational use case, two junction tables, Reaction-input and Reaction-output, are required to model these many-to-many relationships (Fig 1A). Each junction table contains foreign keys of the Reactions and the associated PEs. The SQL query to retrieve input and output entities of a given reaction requires two join operations per junction table (Fig 1B). In the first stage of its execution, each join operation forms the cartesian product between the tables and, during the filtering process, all rows of the result set that are not of interest are discarded.

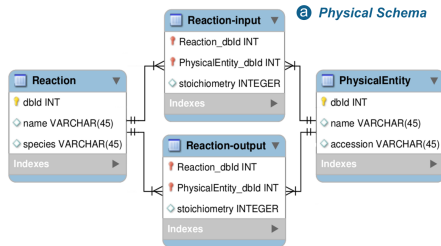
The same structure of a reaction with inputs and outputs can be modelled in a simpler way with Neo4j as exemplified by the reaction presented in Fig 1C. The reaction (green node), contains named outgoing relationships to corresponding input and output entities (purple nodes). Taking advantage of Cypher, the same query, can be written in a shorter but more intuitive manner thanks to its ASCII-Art syntax [3] to represent patterns (Fig 1D). The query describes a pattern that includes a Reaction, again identified by its identifier, with its outgoing input and output relationships. Finally, all nodes matching the specified pattern are returned.

Since their introduction in the 1970's, relational database engines have been optimised to provide efficient execution of SQL queries. This is particularly the case with global queries that aggregate large amounts of data without the need to perform any traversal operations. However, Reactome data contain many relationships, like those illustrated in Fig 1, and thus many join tables, so queries generally require traversal operations, a computational intensive task that tends to result in poor performance compared to graph databases [16]. To address this issue and improve query performance, some resources have created redundant denormalised copies of their relational database [17, 18, 19]. Nowadays, graph databases, such as Neo4j, offer a more appropriate alternative for cases of highly interconnected data.

The new graph database ecosystem

The graph database batch importer (<https://github.com/reactome/graph-importer>) was developed to migrate the content from the relational database used in curation, to a graph database

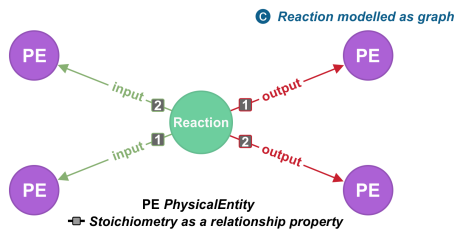
Relational Database



```
SELECT r.*, pe.* FROM Reaction r
JOIN Reaction-input ri ON r.dbId = ri.Reaction_dbId
JOIN PhysicalEntity pe ON pe.dbId = ri.PhysicalEntity_dbId
JOIN Reaction-output ro ON r.dbId = ro.Reaction_dbId
JOIN pe ON pe.dbId = ro.PhysicalEntity_dbId
WHERE r.dbId = IDENTIFIER
```

b SQL query

Graph Database



```
MATCH (r:Reaction{dbId:IDENTIFIER})-[:input|output]->(pe)
RETURN r,pe
```

d Cypher query

Fig 1. A simplified example where reactions only contain reactants and products represented by the class PhysicalEntity. (a) In the relational use case, two junction tables are required to model these many-to-many relationships (b) SQL query used to retrieve input and output entities of a given reaction where two join operations are needed per junction table. (c) The same reaction modelled as a graph. The reaction (green node) contains named outgoing relationships to corresponding input and output entities (purple nodes). (d) The same query written in Cypher, in a shorter but more intuitive manner.

<https://doi.org/10.1371/journal.pcbi.1005968.g001>

during each quarterly release process. Although the underlying data storage was changed, the original data model used by MySQL was kept the same. The conversion was done following a depth-first approach starting from the top level pathways and traversing all the content, ensuring that each object is processed only once during the conversion. Every object constitutes a node in the graph and the edges that connect the nodes correspond to the names of the slots as defined in the domain model (Fig 2). As a result, a Neo4j graph database is generated and contains all the Reactome data. It can be directly used for third parties in order to use Cypher to retrieve the target data.

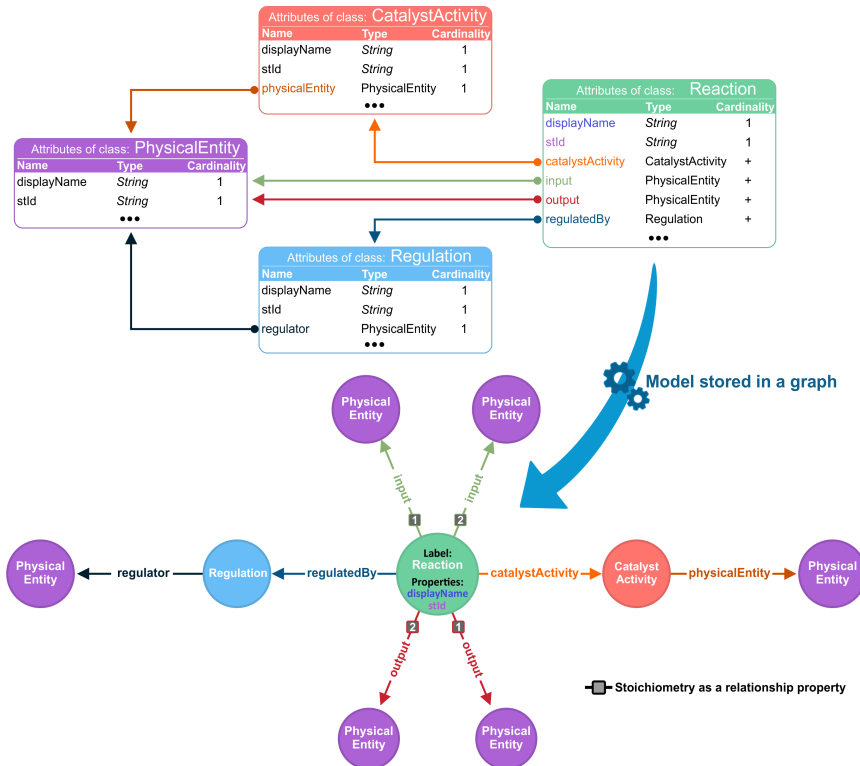


Fig 2. Representation of the content migration. The example shows a Reaction class reduced to its inputs, outputs, catalyst and regulators. A model class instance is converted to a graph database node where (1) slots with primitive value types become node properties and (2) slots allocating instances of another class become relationships.

<https://doi.org/10.1371/journal.pcbi.1005968.g002>

A number of integrity tests have been put in place to ensure that both the graph and relational database have the same content after conversion. These tests are part of the graph-core and they are executed after migrating the relational database to the graph database to ensure that the data has been properly stored. The tests include checks to verify: that the number of top level pathways present in the graph database corresponds to the number of those present in the relational database; that a given pathway in the graph database has the same ancestors as its counterpart in the relational database; that the content of a given complex is the same in both databases.

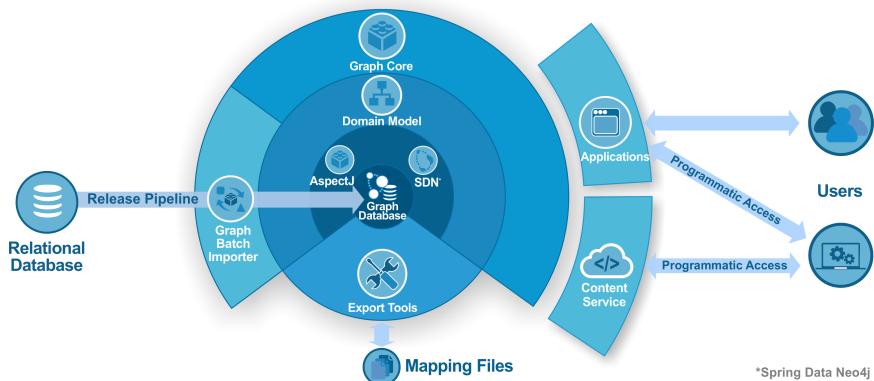


Fig 3. A schematic diagram of the new ecosystem. The relational database is converted to a graph database via the batch importer that relies on the Domain Model. Spring Data Neo4j and AspectJ are two main pillars for the graph-core, which also rests on the Domain Model. Users access services or use tools that make direct use of the graph-core as a library that eliminates the code boilerplate for data retrieval and offers a data persistency mechanism. Finally, export tools take advantage of Cypher to generate flat mapping files.

<https://doi.org/10.1371/journal.pcbi.1005968.g003>

Fig 3 presents a schematic illustration of the new Reactome graph database ecosystem. A library called graph-core (<https://github.com/reactome/graph-core>) was developed on top of the graph database to serve as a data access layer. The aim of the library is to provide easy access and data persistence as well as to reduce the boilerplate code in third party projects that require accessing and traversing Reactome content. The graph-core uses Spring Data Neo4j (SDN) [20] to access the graph content and AspectJ to enable lazy loading [21]. Lazy loading commonly refers to a design pattern, that postpones the retrieval of object attributes until the point at which they are needed. In our case, AspectJ weaver is used to intercept the getter methods and run specific code to silently retrieve more data when needed.

The ContentService (<https://reactome.org/ContentService>) is a REST based web service [22], built on top of the graph-core, to provide programmatic access to the Graph Database for third party developers (<https://github.com/reactome/content-service>). Implemented on top of Spring MVC (<https://spring.io/>), the ContentService utilises the graph-core library and is fully documented with Open API (<https://www.openapis.org/>).

Results and discussion

Among its main advantages, this new solution is faster and less computationally intensive than the previous one based on the relational database. Performing queries against the graph database constitutes a more scalable approach, resulting in higher throughput and, ultimately, to a more robust ContentService able to cope with an always increasing number of requests. Additionally, the resulting product is easier to maintain as most new methods can be added by simply writing the respective Cypher queries, avoiding writing complex algorithms in a given programming language (Fig 1B).

The use cases above are available as methods in the ContentService API (<https://reactome.org/ContentService/>). Fig 4 emphasises how queries of Reactome data have been simplified by the adoption of the graph database. The query in Fig 4A shows how to retrieve the participating molecules for a pathway. The reverse query, identifying pathways where a molecule participates, is shown in Fig 4B, which follows a similar pattern to Fig 4A, but fixes the end-bound and leaves the upper-side open for traversing results. Based on feedback provided by people contacting our help desk (help@reactome.org) and attending our training sessions, the new way of querying Reactome is easy and intuitive to learn, and researchers, who are interested in performing queries against Reactome data, can learn to write them in Cypher in a relatively short amount of time.

To assess the improvement we designed a set of stress tests to measure the impact of adopting the graph database in Reactome. All stress tests were executed on a standard laptop featuring an Intel Core i7 at 2.6 GHz, 16 GB of DDR3 memory at 1,600 MHz, and 256 GB of flash storage. The tests do not aim to compare the two storage technologies (MySQL and Neo4j) but instead their usage by Reactome. The stress tests were run against the web services built on top of each storage technology and included two scenarios: (1) simulation of one user sequentially querying 5,000 reactions for *Homo sapiens* and (2) simulating an increasing set of users simultaneously performing the previous task. In each case the resulting data for every reaction had to be marshalled as an instance of the correspondent model class. The test comprised four executions; two against the previous web service running on top of the relational database and the other two accessing the new web service running on top of the graph database through the newly created graph-core library (<https://github.com/reactome/graph-core>). The reactions were accessed in a sequential fashion to ensure that caching did not provide any sort of advantage for any of the approaches, because a queried object would never be retrieved again in the same test. It should be mentioned that prior to any stress test's execution, both Neo4j and MySQL databases were configured to allocate 50% of the available physical memory (8GB).

As illustrated in Fig 5, querying the data stored in the relational database resulted in significantly longer response times. In particular, in the case of the relational implementation of the Reactome knowledgebase the average query time was 173.11 ms (± 25.81) while in the case of the graph implementation, the average response time dropped to 12.56 ms (± 2.94), a 93%

a. Which molecules participate in Interleukin-4 and 13 signaling (R-HSA-6785807)?

```
MATCH (p:Pathway{stId:"R-HSA-6785807"})-[:hasEvent*]->(rle:ReactionLikeEvent),
      (rle)-[:input|output|catalystActivity|entityFunctionalStatus|physicalEntity|regulatedBy|
            regulator|hasComponent|hasMember|hasCandidate|repeatedUnit*]->(pe:PhysicalEntity),
      (pe)-[:referenceEntity]->(re:ReferenceEntity)-[:referenceDatabase]->(rd:ReferenceDatabase)
RETURN DISTINCT re.identifier AS Identifier, rd.displayName AS Database
```

b. In which pathways does CCR5 (UniProt:P51681) participate?

```
MATCH (p:Pathway)-[:hasEvent*]->(rle:ReactionLikeEvent),
      (rle)-[:input|output|catalystActivity|entityFunctionalStatus|physicalEntity|regulatedBy|
            regulator|hasComponent|hasMember|hasCandidate|repeatedUnit*]->(pe:PhysicalEntity),
      (pe)-[:referenceEntity]->(re:ReferenceEntity{identifier:"P51681"}),
      (re)-[:referenceDatabase]->(rd:ReferenceDatabase{displayName:"UniProt"})
RETURN DISTINCT p.stId AS Identifier, p.displayName AS Pathway
```

Fig 4. Examples of frequent use cases that can be answered using Cypher queries. a) Retrieving the participating molecules for "Interleukin-4 and 13 signalling" pathway. b) Retrieving the pathways in which CCR5 participates.

<https://doi.org/10.1371/journal.pcbi.1005968.g004>

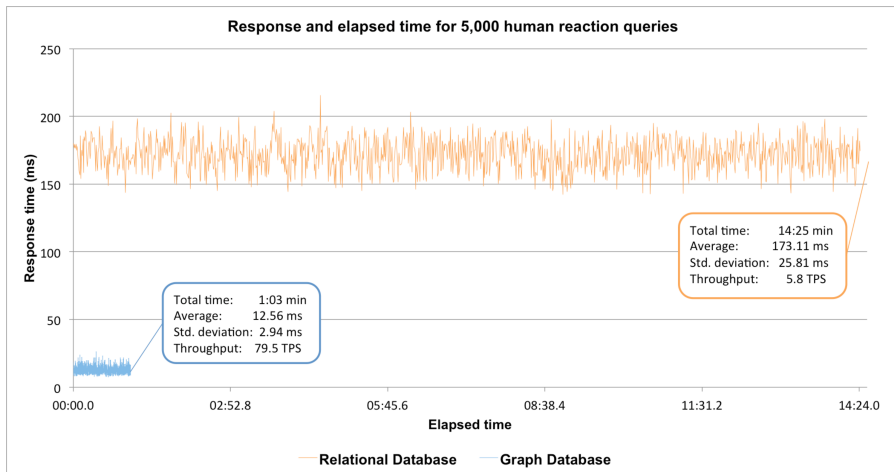


Fig 5. Comparison of the response and elapsed time for one user sequentially retrieving 5,000 reaction instances from the graph and relational databases (blue and orange respectively). The graph database software ecosystem achieved a 93% average improvement in performance compared to that of the relational database.

<https://doi.org/10.1371/journal.pcbi.1005968.g005>

reduction in the average query time. The new implementation supported higher throughput, in terms of transactions per second (TPS), reaching 79.5 TPS compared to 5.8 TPS. As a result of this boost in performance, all 5,000 queries to the graph database were performed in 63 seconds while the relational implementation required more than 14 minutes for the same task.

A second stress test simulated a more realistic scenario where multiple users perform concurrent database queries (Fig 6). Once again, querying the Reactome knowledgebase in its relational implementation resulted in significantly longer response times. For instance, in case of 10 concurrent threads performing queries to the relational implementation of the Reactome knowledgebase the average response time was 1,516 ms while in the case of the graph implementation, the average response time dropped to 49.05 ms. In addition, the new implementation achieved higher throughput reaching 203.6 TPS compared to 6.6 TPS. Consequently, the graph implementation of Reactome provides higher scalability enabling Reactome to handle larger volumes of user requests.

Fig 7 presents a comparison between the throughputs achieved by both systems against the number of users performing concurrent queries. The graph implementation achieved a higher number of transactions per second that reached a plateau after the point where the number of active threads becomes equal to the available processor cores; in this case 4. On the other hand, the measured throughput in case of the relational implementation is stable and does not seem to take advantage of any concurrency.

Many users choose to download the Reactome graph database and access the data through Cypher queries directly in their computers. Our usage statistics show that a growing number of users have downloaded the Reactome graph database and, based on the questions gathered by our help desk service, we believe that they have used it to perform local queries against the

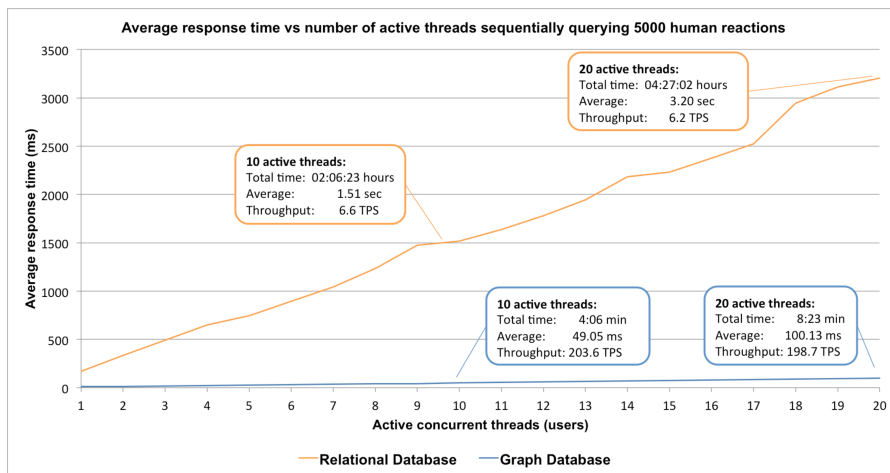


Fig 6. Response time versus an increasing set of users simultaneously performing queries for 5,000 reaction instances. Starting with one and scaling up to 20 concurrent users, the relational database performance drops while the graph database keeps a low response time and a good throughput as the number of active threads increases.

<https://doi.org/10.1371/journal.pcbi.1005968.g006>

complete Reactome knowledgebase. In particular, during the first year that Reactome provided the graph database, there were 2,385 downloads by 912 unique users. 118 of those users downloaded the graph database after each data release. It is worth mentioning that during writing of this manuscript, the size of the Reactome relational database in its current data release (v62) is around 2.0GB while the size of the graph database is approximately 1.8GB. Fig 8 provides a summary of the graph database.

With a tool so powerful at managing highly connected data sets and complex queries at our disposal, Reactome is providing faster and more stable services to researchers around the world. In the near future, Reactome plans to upgrade its services and leverage the full potential of Cypher in order to provide answers to questions that require diving deeper into our data. In particular, the integration of a graph database lowers the complexity of problems that require traversing our knowledgebase, such as identifying causal interactions or revealing all possible paths between two molecules.

Future development in Reactome is not likely to be affected by the fact that Neo4j is by nature schema-less, mainly because the rigid schema of our relational database with all the applied constraints is used to ensure data consistency during the curation phase. Currently, data are migrated to Neo4j during each quarterly release process and are used to speed up queries in production.

In conclusion, through the adoption of the Neo4j graph database, and by harnessing the power of its query language, Reactome provides efficient access to its pathway knowledgebase. As a result of this shift in the underlying data storage technology, the average query time has been reduced up to 93%. In addition, the graph-core library and the ContentService leverage

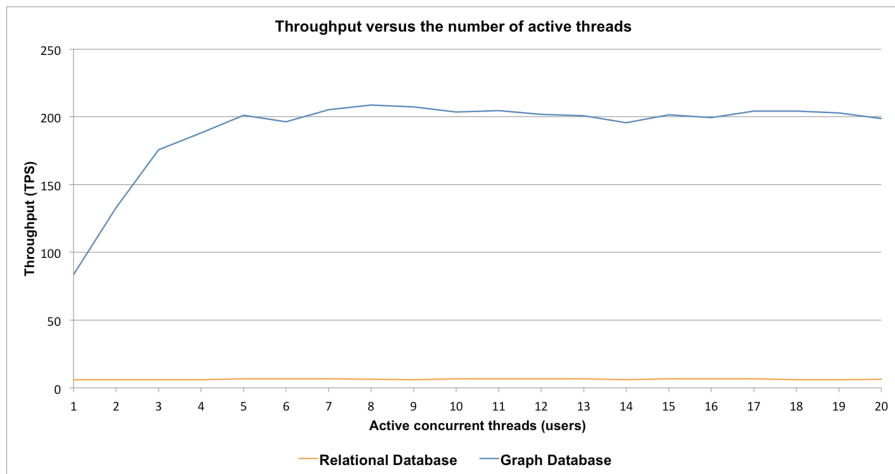


Fig 7. Throughput measured in transactions per second, versus the number of users concurrently performing queries for 5,000 reaction instances in *Homo sapiens*.
<https://doi.org/10.1371/journal.pcbi.1005968.g007>

these benefits of this shift and can be used by third party applications to efficiently access Reactome.

Reactome's successful use case constitutes a strong argument in favour of the positive impact this new technology can have in the field. By following Reactome's use case, other community projects with similar complex models could benefit from moving their storage to a graph database while keeping their data model. While we have demonstrated the major impact of moving the Reactome public database to a graph database in terms of usability, stability, and response time, we think this is only a milestone in the growing ecosystem of network-oriented biomolecular data resources that will enable entirely new functionalities through moving to modern database technology that better reflects the graph-like structure of their source data. While we will work directly with internal and external resources to move along that path, we would also like to invite the community to use the open data Reactome graph database to develop their own novel uses of Reactome data.

Availability and future directions

The Reactome graph database is freely available at: <https://reactome.org/dev/graph-database>. The API for the ContentService is available at <https://reactome.org/ContentService> with documentation and tutorials available at: <https://reactome.org/dev/content-service>. The source code, in Java, is freely available at: <https://github.com/reactome> (See the graph-core, graph-importer and content-service repositories).

Future development will focus on updating the version of SDN and integrating interaction data from IntAct (<http://www.ebi.ac.uk/intact/>) directly to the Reactome graph database.

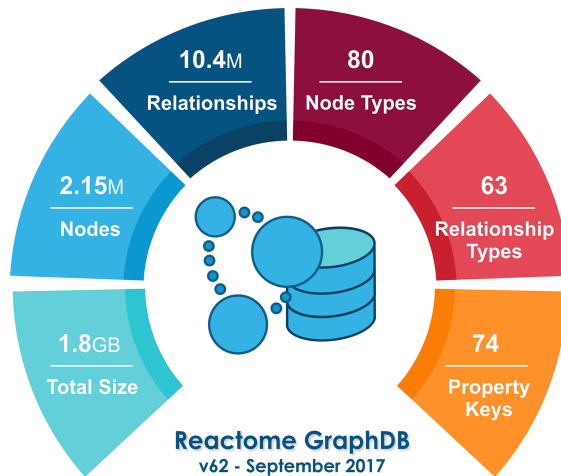


Fig 8. The Reactome graph database in numbers.

<https://doi.org/10.1371/journal.pcbi.1005968.g008>

Author Contributions

Funding acquisition: Henning Hermjakob.

Investigation: Antonio Fabregat.

Methodology: Antonio Fabregat.

Software: Antonio Fabregat, Florian Korninger, Guilherme Viteri.

Supervision: Antonio Fabregat.

Validation: Antonio Fabregat.

Writing – original draft: Antonio Fabregat, Konstantinos Sidiropoulos.

Writing – review & editing: Antonio Fabregat, Konstantinos Sidiropoulos, Pablo Marin-Garcia, Peipei Ping, Guanming Wu, Lincoln Stein, Peter D'Eustachio, Henning Hermjakob.

References

1. Fabregat A, Sidiropoulos K, Garapati P, Gillespie M, Hausmann K, Haw R, et al. The Reactome pathway Knowledgebase. *Nucleic Acids Res.* 2016; 44:D481–7. <https://doi.org/10.1093/nar/gkv1351> PMID: 26656494
2. Van Bruggen R. *Learning Neo4j*. Birmingham: Packt Publishing Ltd.; 2014
3. Vukotic A, Watt N, Abedrabbo T, Fox D, Partner J. *Neo4j in Action*. 1st ed. Shelter Island, NY: Manning Publications; 2014.
4. Sedgewick R, Wayne K. *Algorithms*. 4th ed. Addison-Wesley; 2011. pp. 566–596.

5. Vastrik I, D'Eustachio P, Schmidt E, Joshi-Tope G, Gopinath G, Croft D, et al. Reactome: a knowledge base of biologic pathways and processes. *Genome Biol.* 2007; 8: R39. <https://doi.org/10.1186/gb-2007-8-3-r39> PMID: 17367534
6. Have CT, Jensen LJ. Are graph databases ready for bioinformatics? *Bioinformatics.* 2013; 29(24):3107. <https://doi.org/10.1093/bioinformatics/btt549> PMID: 24135261
7. Henkel R, Wolkenhauer O, Waltemath D. Combining computational models, semantic annotations and simulation experiments in a graph database. *Database (Oxford).* 2015; 8: 2015.
8. Summer G, Kelder T, Ono K, Radonjic M, Heymans S, Demchak B. cyNeo4j: connecting Neo4j and Cytoscape. *Bioinformatics.* 2015; 31(23):3868–9. <https://doi.org/10.1093/bioinformatics/btv460> PMID: 26272981
9. Lysenko A, Roznovat IA, Saqi M, Mazein A, Rawlings CJ, Auffray C. Representing and querying disease networks using graph databases. *BioData Min.* 2016; 9(1):23.
10. Toure V, Mazein A, Waltemath D, Balaur I, Saqi M, Henkel R, et al. STON: exploring biological pathways using the SBGN standard and graph databases. *BMC Bioinformatics.* 2016; 17: 494. <https://doi.org/10.1186/s12859-016-1394-x> PMID: 27919219
11. Balaur I, Mazein A, Saqi M, Lysenko A, Rawlings CJ, Auffray C. Recon2Neo4j: applying graph database technologies for managing comprehensive genome-scale networks. *Bioinformatics.* 2016;
12. Swainston N, Batista-Navarro R, Carbonell P, Dobson PD, Dunstan M, Jervis AJ, et al. biochem4j: Integrated and extensible biochemical knowledge through graph databases. *PLoS ONE.* 2017; 12(7): e0179130. <https://doi.org/10.1371/journal.pone.0179130> PMID: 28708831
13. Robinson I, Webber J, Eifrem E. *Graph Databases.* O'Reilly Media, Incorporated; 2013.
14. Lal M. *Neo4j graph data modeling.* Birmingham: Packt Publishing Ltd.; 2015.
15. Neubauer P. (2010) *Graph Databases, NOSQL and Neo4j.* InfoQ. 12 May 2010. Available from: <https://www.infoq.com/articles/graph-nosql-neo4j>. Cited 20 June 2017.
16. Vicknair C, Macias M, Zhao Z, Nan X, Chen Y, Wilkins D. A comparison of a graph database and a relational database: a data provenance perspective. *Proceedings of the 48th Annual Southeast Regional Conference*; 2010 Apr 15–17; New York, NY; 2010.
17. Birney E, Andrews D, Bevan P, Caccamo M, Cameron G, Chen Y, et al. Ensembl2004. *Nucleic Acids Res.* 2014; 32:D468–D470.
18. Eppig JT, Blake JA, Bult CJ, Kadin JA, Richardson JE, Mouse Genome Database Group. The Mouse Genome Database (MGD): facilitating mouse as a model for human biology and disease. *Nucleic Acids Res.* 2015; 43:D726–D736 <https://doi.org/10.1093/nar/gku967> PMID: 25348401
19. Štefanić S, Lexa M. A Flexible Denormalization Technique for Data Analysis above a Deeply-Structured Relational Database: Biomedical Applications. In: Ortuño F, Rojas I, editor. *Bioinformatics and Biomedical Engineering. IWBBIO 2015. Lecture Notes in Computer Science.* Springer, Cham; 2015. vol 9043.
20. Hunger M. *Good Relationships: The Spring Data Neo4j Guide Book.* InfoQ enterprise software development, C4Media; 2012.
21. Laddad R. *AspectJ in Action.* 2nd ed. Manning; 2010.
22. World Wide Web Consortium (W3C) *Web Services Architecture, W3C Working Group Note.* 11 February 2004. Available from: <https://www.w3.org/TR/ws-arch/>. Cited 20 June 2017.

P.5. THE REACTOME PATHWAY KNOWLEDGEBASE (NAR 2017)

Published online 14 November 2017

Nucleic Acids Research, 2018, Vol. 46, Database issue D649–D655
doi: 10.1093/nar/gkx1132

The Reactome Pathway Knowledgebase

Antonio Fabregat^{1,2,†}, Steven Jupe^{1,†}, Lisa Matthews^{3,†}, Konstantinos Sidiropoulos¹, Marc Gillespie^{4,5}, Phani Garapati¹, Robin Haw⁴, Bijay Jassal⁴, Florian Korninger¹, Bruce May⁴, Marija Milacic⁴, Corina Duenas Roca¹, Karen Rothfels⁴, Cristoffer Sevilla¹, Veronica Shamovsky³, Solomon Shorser⁴, Thawfeek Varusai¹, Guilherme Viteri¹, Joel Weiser⁴, Guanming Wu^{6,*}, Lincoln Stein^{4,7,*}, Henning Hermjakob^{1,8,*} and Peter D'Eustachio^{3,*}

¹European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, Cambridgeshire CB10 1SD, UK, ²Open Targets, Wellcome Genome Campus, Hinxton, Cambridgeshire CB10 1SD, UK, ³NYU School of Medicine, New York, NY 10016, USA, ⁴Ontario Institute for Cancer Research, Toronto, ON, M5G 0A3, Canada, ⁵College of Pharmacy and Health Sciences, St. John's University, Queens, NY 11439, USA, ⁶Oregon Health Sciences University, Portland, OR 97239, USA, ⁷Department of Molecular Genetics, University of Toronto, Toronto, ON M5S 1A1, Canada and ⁸National Center for Protein Sciences, Beijing, China

Received October 07, 2017; Revised October 20, 2017; Editorial Decision October 21, 2017; Accepted October 26, 2017

ABSTRACT

The Reactome Knowledgebase (<https://reactome.org>) provides molecular details of signal transduction, transport, DNA replication, metabolism, and other cellular processes as an ordered network of molecular transformations—an extended version of a classic metabolic map, in a single consistent data model. Reactome functions both as an archive of biological processes and as a tool for discovering unexpected functional relationships in data such as gene expression profiles or somatic mutation catalogues from tumor cells. To support the continued brisk growth in the size and complexity of Reactome, we have implemented a graph database, improved performance of data analysis tools, and designed new data structures and strategies to boost diagram viewer performance. To make our website more accessible to human users, we have improved pathway display and navigation by implementing interactive Enhanced High Level Diagrams (EHLs) with an associated icon library, and subpathway highlighting and zooming, in a simplified and reorganized web site with adaptive design. To encourage re-use of our content, we have enabled export of pathway diagrams as 'PowerPoint' files.

INTRODUCTION

At the cellular level, life is a network of molecular reactions that include signal transduction, transport, DNA replication, protein synthesis, and intermediary metabolism. A variety of online resources capture aspects of this information at the level of individual reactions such as Rhea (1) or at the level of reaction sequences spanning various domains of biology such as KEGG (2), MetaCyc (3) or PANTHER (4). The Reactome Knowledgebase is distinctive in focusing its manual annotation effort on a single species, *Homo sapiens*, and applying a single consistent data model across all of these domains of biology. Processes are systematically described in molecular detail to generate an ordered network of molecular transformations, resulting in an extended version of a classic metabolic map (5). The Reactome Knowledgebase systematically links human proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for discovering unexpected functional relationships in data such as gene expression surveys or catalogs of somatic mutations in tumor cells.

Reactome (version 62—September 2017) has entries for 10 719 human genes, 53% of the 20 338 predicted human protein-coding genes (http://www.ensembl.org/Homo_sapiens/Info/Annotation), supporting the annotation of 24 704 specific forms of proteins distinguished by co- and post-translational modifications and subcellu-

[†]To whom correspondence should be addressed. Tel: +1 212 263 5779; Fax: +1 212 263 8166; Email: deustp01@med.nyu.edu
Correspondence may also be addressed to Henning Hermjakob. Tel: +44 1223 494 671; Email: hhe@ebi.ac.uk
Correspondence may also be addressed to Lincoln Stein. Tel: +1 416 673 8514; Email: Lincoln.Stein@oicr.on.ca
Correspondence may also be addressed to Guanming Wu. Tel: +1 503 494 4502; Fax: +1 503 346 6815; Email: wug@ohsu.edu
^{*}These authors contributed equally to this work as first authors.

D650 Nucleic Acids Research, 2018, Vol. 46, Database issue

lar localizations. These function with 1768 small molecules as substrates, catalysts, and regulators in 11 302 reactions annotated on the basis of data from 27 526 literature references. These tallies include 1334 mutant variants and their post-translationally modified forms derived from 285 gene products, used to annotate 906 disease-specific reactions, tagged with 294 Disease Ontology terms (6). These reactions form 2102 pathways (e.g. Interleukin-15 signaling; phosphatidylinositol phosphate metabolism; receptor-mediated mitophagy) grouped into 26 superpathways that correspond to domains of biology such as metabolism and signal transduction. Reactome's dataset continues to grow briskly, with 74 new human pathways added in the first three quarters of 2017.

Notable additions include extensive new annotations of cytokine signaling, including a comprehensive catalog of known interleukin signaling pathways. We have also revised and supplemented existing pathways, continuing to build our catalogs of signaling processes mediated by G protein-coupled receptors, of transport processes, and of metabolism. Notably, where our initial annotations in these domains centered on the most extensively studied, 'text-book' versions of pathways and molecules, we are now systematically adding proteins whose properties indicate closely-related biological roles, to increase the density and connectivity of the reaction network in Reactome that is available for visualization and computational analysis.

This growth has come at a cost. Our SBGN-based (7) scheme for representing pathways, implemented eight years ago, yields pathway diagrams that become cluttered and difficult for biologist users to navigate as we achieve more nearly complete annotations of the participants in a process and their functions, and our approximately 120 diagrams of superpathways are essentially lists of the names of component pathways, functional but uninformative and unappealing to biologist users. Meanwhile, this growth in the number and complexity of our annotations has made our relational data structure slow and unwieldy for handling complex queries and large scale data analyses.

We have addressed the computational aspects of these challenges by implementing a Neo4j graph database structure for our production web site, developing a new high-performance in-memory implementation of our core over-representation data analysis tool, and implementing a new Pathway Diagram Viewer to support faster data loading, diagram rendering and element seeking. To improve usability we have developed Enhanced High Level Diagrams (EHLDs) that combine an iconography familiar from textbooks and review articles with web functionality, to represent superpathways, and have added features to our pathway diagrams to improve their legibility. A redesigned web site with adaptive technology is accessible from tablets and mobile devices, and supports more intuitive navigation.

Here, we provide brief descriptions of these changes and their contributions to the usability of the Reactome data resources.

IMPROVED PERFORMANCE AND SCALABILITY

Implementation of a graph database

Relational databases work well to model and store complex pathway information and their engines have been optimized to provide efficient execution of global SQL queries that aggregate large amounts of data without requiring traversal operations. Reactome data, however, contain many relationships and thus many join tables, so queries generally require traversal operations resulting in degraded performance and long response times.

To preserve our well-established and well-tested tools for data annotation and internal storage while improving performance of our public resource, we continue to maintain the relational Reactome database but have developed a graph database batch importer to migrate content to a Neo4j graph database during each quarterly release. Third parties can directly use Cypher, a declarative, pattern matching query language specifically designed to retrieve data from graph data structures. A fully documented REST based web content service has been built on top of the graph-core to provide third party developers with programmatic access to the graph database. Stress-testing the graph and relational implementations of Reactome indicates approximately a 30-fold improvement in throughput for queries submitted to the graph implementation. Users can download the Reactome graph database and access the data through Cypher queries directly on their computers. In the first year that Reactome provided the graph database, there were 2385 downloads by 912 unique users, 118 of whom downloaded the graph database after each data release.

The Reactome graph database is freely available at: <https://reactome.org/dev/graph-database>. The API for the ContentService is available at <https://reactome.org/ContentService> with documentation and tutorials available at: <https://reactome.org/dev/content-service>. The Java source code is freely available at: <https://github.com/reactome> in the graph-core, graph-importer and content-service repositories. (See Table 1 for a summary of on-line resources discussed in this paper.)

Improved performance of data analysis tools

Reactome provides an over-representation analysis tool (8) to support interpretation of expression data sets. The tool calculates whether a user-generated list of proteins, for example, ones whose expression is changed in response to a stress, contains more annotated to each Reactome pathway than would be expected by chance given the number of proteins in the set, the number annotated to the pathway, and the number annotated in all of Reactome. The Reactome implementation uses a hypergeometric distribution test to generate a probability score, which is corrected for false discovery rate using the Benjamani-Hochberg method (9,10). We have now developed a high-performance in-memory implementation that divides the method into four steps, each with a specific data structure to improve performance and minimize the memory footprint. First, each identifier in the user's sample is matched to an entity in Reactome using a radix tree as a lookup table. Second, a graph is used to

Table 1. Reactome online resources

Home page	https://reactome.org
an introductory video for users	https://youtu.be/skixrvI4nU
The Reactome graph database	https://reactome.org/dev/graph-database
API for the ContentService	https://reactome.org/ContentService
documentation and tutorials	https://reactome.org/dev/content-service
Java source code	https://github.com/reactome
includes graph-core, graph-importer, content-service and analysis-tools repositories	
EHLD source code	https://github.com/reactome-pwp/diagram
reusable stand-alone JavaScript EHLD viewer	https://reactome.org/dev/diagram/js
EHLD icon library	https://reactome.org/icon-lib
community contributions to the library	https://reactome.org/icon-info

model proteins, chemicals, their orthologs in other species and their composition in complexes and sets. The third and fourth steps aggregate the results and calculate the statistics, employing a double-linked tree. This implementation provides a stable, high performance pathway analysis service, enabling the analysis of genome-wide datasets within seconds, allowing interactive exploration and analysis of high throughput data. It is accessible on our web site both via the AnalysisService for programmatic access and a user submission interface integrated into the PathwayBrowser. All of its source code is freely available in the Analysis-Tools repository in the Reactome GitHub (<https://github.com/reactome/>) (10).

Data structures and strategies to boost diagram viewer performance

A new version of the Pathway Diagram Viewer (version 3) was implemented to provide faster data loading, diagram rendering and element seeking, with the goal of completing most user interactions in less than a second. Improvements include: (i) restructuring of the data format used to send the data from the server to the client from XML to JSON, (ii) using a graph data structure to store the pathway content on the client side, (iii) boosting the client content load strategy, (iv) implementing a multi-layer canvas approach and (v) utilizing a space partitioning data structure to store the elements to be rendered. Conversion from XML to JSON reduced diagram sizes only by about 20% but reduced processing and loading times by at least 65% over the full range of sizes.

IMPROVED PATHWAY DISPLAY AND NAVIGATION

Implementation of interactive Enhanced High Level Diagrams (EHLDS)

To improve the quality of the graphics used to represent pathways and to make pathway navigation easier, we have integrated three new features into the Reactome pathway browser: textbook-style EHLDS to represent superpathways such as signaling and immune function, a mechanism to highlight different subpathways with colored overlays in zoomed-out views of detailed pathway diagrams coupled to changes in amount of detail displayed as users zoom into and out of diagrams, and an option to export EHLDS, individual EHLD icons and pathway diagrams in editable, reusable forms (11).

EHLDS (Figure 1) resemble overviews of biological processes shown in textbooks or review articles, with a consistent iconography based on widely used simple diagrams of molecules, cellular structures, cells, and tissues, to make navigation intuitive and familiar. This design helps the user to recognize the process represented in an EHLD and select the appropriate region of the EHLD to navigate to the next more detailed level of the Reactome hierarchy, ultimately to arrive at a detailed pathway diagram that shows individual physical entities participating in individual reactions.

EHLDS were produced by a team of curators and illustrators who revised the Reactome event hierarchy as necessary and developed overview diagrams for the higher levels of the hierarchy. They developed a common style and iconography to enhance the 'recognition effect' and facilitate efficient navigation. Of the 120 high-level pathways represented as lists of subpathway names, 48 have been converted to EHLDS; conversion of the remainder should be complete by late in 2018.

The SVG format was selected for EHLDS because it allows object-based vector representation for easy editing, resolution-independent zooming, interaction features for richer interfaces, use of data in regular text format as a subset of an eXtensible Markup Language (XML) that can be easily handled by computer programs or text editors, and support by most popular web browsers and compatibility with most popular graphic software packages.

EHLDS have a limited number of entities, so the multilayered HTML5 Canvas approach implemented for detailed pathway diagrams (above) was not needed. Instead, for EHLDS, their simple overlay features and a rendering strategy where the flow of information does not depend on the level of zoom, allowed the adoption of an SVG renderer. This allowed developers to implement features such as overlays, zoom and translation by applying a series of filters and transformations.

The Reactome web interface now has an SVG rendering component that, beyond standard zooming and panning, allows specific regions of an EHLD to be highlighted, recognizes mouse click events on specific regions to allow navigation to subpathways and allows analysis results to be overlaid on it. All of the required technical annotations were included in the SVG files by using the database identifier attribute of each relevant diagram element. Any element with no technical annotation was considered a decoration, so EHLDS can accommodate placeholder icons for subpathways that are still in development. The code is part of the EHLD package available on the Reactome pub-

D652 *Nucleic Acids Research*, 2018, Vol. 46, Database issue

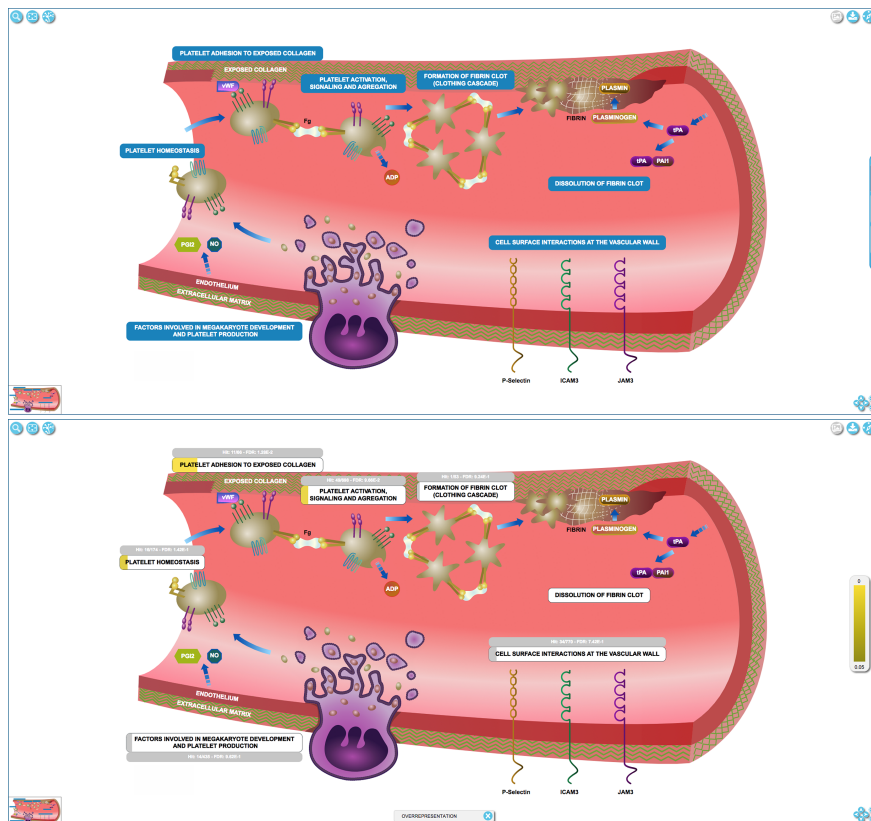


Figure 1. 'Hemostasis' top-level pathway represented as an Entity High Level Diagram (EHL) above, and with expression data analysis results overlaid, below, showing relative overexpression of gene products involved in platelet adhesion to exposed collagen.

lic GitHub repository (<https://github.com/reactome-pwp/diagram>). EHLs are also available in the reusable standalone JavaScript diagram viewer (<https://reactome.org/dev/diagram.js>).

Pathway analysis results (1) can be overlaid on EHLs. The results are displayed in the label of each subpathway, which is overlaid by a colored rectangular shape whose width and color represent the percentage of hit entities and the *P*-value, respectively (Figure 1).

EHLs, including any analysis result overlay, can be easily downloaded and saved in SVG format. Users can edit the content of the exported files with commercial or open-source graphics applications. EHLs use a consistent

iconography that reuses glyphs when the same entity plays a role in more than one biological process. We have made a library of these graphical elements to provide them in SVG, PNG and EMF formats. The icon library is available at <https://reactome.org/icon-lib> and is distributed under the terms of the CC-BY license (<https://creativecommons.org/licenses/by/4.0/>) to facilitate the creation of uniform diagrams through the use of pre-existing glyphs and to offer these components to the community for reuse.

Researchers can use these icons to create their own illustrations to convey their findings and ideas, whether in a paper or a grant application. Additionally, we encourage the

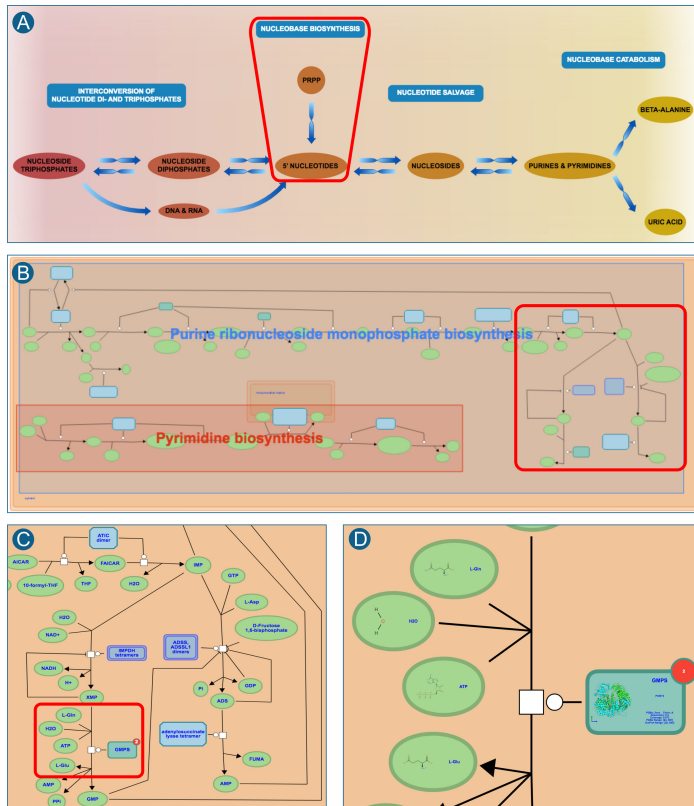


Figure 2. Pathway navigation with detail matched to zoom level. A user who selects the pathway 'nucleotide metabolism' is presented with an EHL that shows the entire process with its major subpathways labeled (A). Double-clicking anywhere in the region of nucleobase biosynthesis (red box in A) yields a pathway diagram for that process, with its subpathways labeled and only its major components shown (B). As the user zooms in to view the last steps of purine biosynthesis (red box in B), housekeeping entities are revealed and names of all entities are displayed (C). Finally, as the user zooms in to a specific region of the pathway (red box in C), structures of small molecules and proteins are shown (D).

community to contribute new icons to extend this library (<https://reactome.org/icon-info>).

Subpathway highlighting and zooming

To make the navigation of complex SBGN-like pathway diagrams easier, the web display has been modified to make the amount and kind of information displayed dependent on zoom level, as shown in Figure 2. When a user enters a pathway diagram from the event hierarchy or a superpathway EHL, only key components are shown, unlabeled, but with boxes superimposed to highlight pathway bound-

aries and show pathway names. As a user zooms further into the pathway more detail is shown until individual entities are shown as molecular structures. For fast loading and a consistent look and feel, the position and color of pathway boxes is calculated on the server side at database release time.

Export of pathway diagrams as 'PowerPoint' files

Pathway diagrams have previously been available as static PNG images. To enable users to modify diagrams for their own use, we have developed a strategy to export a path-

D654 *Nucleic Acids Research*, 2018, Vol. 46, Database issue

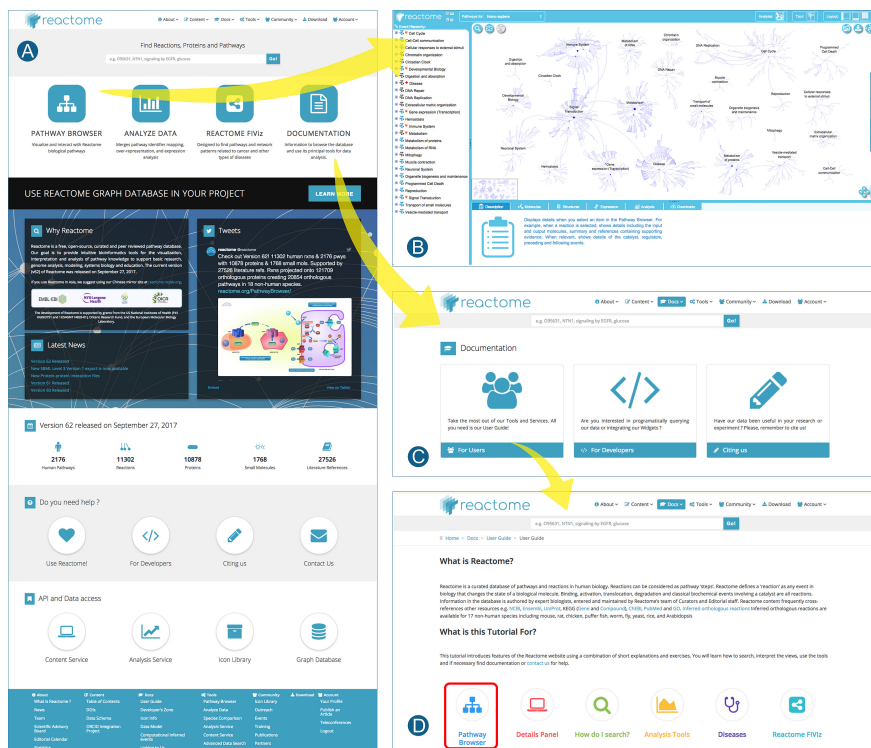


Figure 3. Reorganized Reactome website. From the simplified home page (A), a single step leads to the pathway browser (B), our main tool for data visualization and analysis for human users. Users needing help in navigating and interpreting the site can navigate in a single step to on-line documentation, organized by topic and including solved examples (C). The same documentation button also leads to pages for developers who want access to our analysis service, content service, graph database and widget to include Reactome pathway displays in their web applications (D).

way diagrams from the Pathway Browser as an interconnected set of objects that adapt to layout changes so that when a glyph is moved in an exported diagram all connected objects follow. Our implementation (11) generates PPTX files on the server side, using the color profile selected by the user on the client side and uses Aspose.Slides (<https://www.aspose.com/products/slides/java>), a commercial JAVA API for reading, writing and manipulating PowerPoint documents, to store the content in PPTX format.

A simplified and reorganized web site with adaptive design

We have redesigned our landing page and several key pages linked to it to simplify them and make the navigation to desired tools and documentation more straightforward (Figure 3). The new design includes adaptive features to make

our content accessible from tablets and mobile devices. Documentation for biologist users and developers has been revised to bring it up to date and reorganized to make it more easily navigated.

A video that introduces the pathway browser and data analysis tools is available at <https://youtu.be/skixrv14nU>.

CONCLUSIONS

Both the Reactome Knowledgebase and user needs for data visualization and analysis have grown in size and complexity, and will continue to grow briskly. We have addressed the computational aspects of this growth and change by implementing a Neo4j graph database structure for our production web site, developing a new high-performance in-memory implementation of our core overrepresentation

data analysis tool, and implementing a new Pathway Diagram Viewer to support faster data loading, diagram rendering and element seeking. To improve usability we have developed Enhanced High Level Diagrams (EHLs) that combine an iconography familiar from textbooks and review articles with web functionality, to represent superpathways, and have added features to our pathway diagrams to improve their legibility. A redesigned web site with adaptive technology is accessible from tablets and mobile devices, and supports more intuitive navigation.

ACKNOWLEDGEMENTS

We are grateful to the many expert scientists who have collaborated with us as external authors and reviewers of Reactome content. The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

FUNDING

National Institutes of Health [P41HG003751 and U54GM114833]; European Bioinformatics Institute (EMBL-EBI); Open Targets (The Target Validation Platform); Medicine by Design (University of Toronto). Funding for open access charge: National Institutes of Health [P41HG003751].
Conflict of interest statement. None declared.

REFERENCES

- Morgat,A., Lombardot,T., Axelsen,K.B., Aimo,L., Niknejad,A., Hyka-Nouspikel,N., Couderet,E., Pozzato,M., Pagni,M., Moretti,S. *et al.* (2017) Updates in Rhea – an expert curated resource of biochemical reactions. *Nucleic Acids Res.*, **45**, D415–D418.
- Kanehisa,M., Furumichi,M., Tanabe,M., Sato,Y. and Morishima,K. (2017) KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res.*, **45**, D353–D361.
- Caspi,R., Billington,R., Ferrer,L., Foerster,H., Fulcher,C.A., Keseler,I.M., Kothari,A., Krummenacker,M., Latendresse,M., Mueller,L.A. *et al.* (2016) The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Res.*, **44**, D471–D480.
- Mi,H., Huang,X., Muruganujan,A., Tang,H., Mills,C., Kang,D. and Thomas,P.D. (2017) PANTHER version 11: expanded annotation data from Gene Ontology and Reactome pathways, and data analysis tool enhancements. *Nucleic Acids Res.*, **45**, D183–D189.
- Fabregat,A., Sidiropoulos,K., Garapati,P., Gillespie,M., Hausmann,K., Haw,R., Jassal,B., Jupe,S., Korminger,F., McKay,S. *et al.* (2016) The Reactome Pathway Knowledgebase. *Nucleic Acids Res.*, **44**, D481–D487.
- Kibbe,W.A., Arze,C., Felix,V., Mitraka,E., Bolton,E., Fu,G., Mungall,C.J., Binder,J.X., Malone,J., Vasant,D. *et al.* (2015) Disease Ontology 2015 update: an expanded and updated database of human diseases for linking biomedical knowledge through disease data. *Nucleic Acids Res.*, **43**, D1071–D1078.
- Le Novère,N., Hucka,M., Mi,H., Mo典die,S., Schreiber,F., Sorokin,A., Demir,E., Wegner,K., Aladjem,M.L., Wimalaratne,S.M. *et al.* (2009) The systems biology graphical notation. *Nat. Biotechnol.*, **27**, 735–741.
- Huang,D.W., Sherman,B.T. and Lempicki,R.A. (2009) Survey and summary: Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.*, **37**, 1–13.
- Drăghici,S., Khatri,P., Martins,R.P., Ostermeier,G.C. and Krawetz,S.A. (2003) Global functional profiling of gene expression. *Genomics*, **81**, 98–104.
- Fabregat,A., Sidiropoulos,K., Viteri,G., Forner,O., Marin-Garcia,P., Arnau,V., D'Eustachio,P., Stein,L. and Hermjakob,H. (2016) Reactome pathway analysis: a high-performance in-memory approach. *BMC Bioinformatics*, **18**, 142.
- Sidiropoulos,K., Viteri,G., Sevilla,C., Jupe,S., Webber,M., Orlic-Milacic,M., Jassal,B., May,B., Shamovsky,V., Duenas,C. *et al.* (2017) Reactome enhanced pathway visualization. *Bioinformatics*, <https://doi.org/10.1093/bioinformatics/btx441>.

P.6. THE REACTOME PATHWAY KNOWLEDGEBASE (NAR 2015)

Published online 9 December 2015

Nucleic Acids Research, 2016, Vol. 44, Database issue D481–D487
doi: 10.1093/nar/gkv1351

The Reactome pathway Knowledgebase

Antonio Fabregat¹, Konstantinos Sidiropoulos¹, Phani Garapati¹, Marc Gillespie^{2,3}, Kerstin Hausmann¹, Robin Haw², Bijay Jassal², Steven Jupe¹, Florian Korninger¹, Sheldon McKay², Lisa Matthews⁴, Bruce May², Marija Milacic², Karen Rothfels², Veronica Shomovskiy⁴, Marissa Webber², Joel Weiser², Mark Williams¹, Guanming Wu², Lincoln Stein^{2,5,6,*}, Henning Hermjakob^{1,7,*} and Peter D'Eustachio^{4,*}

¹European Bioinformatics Institute (EMBL-EBI), European Molecular Biology Laboratory, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK, ²Ontario Institute for Cancer Research, Toronto, ON M5G0A3, Canada, ³College of Pharmacy and Health Sciences, St John's University, Queens, NY 11439, USA, ⁴NYU School of Medicine, New York, NY 10016, USA, ⁵Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11724, USA, ⁶Department of Molecular Genetics, University of Toronto, Toronto, ON M5S 1A1, Canada and ⁷National Center for Protein Sciences, Beijing, China

Received October 1, 2015; Revised November 19, 2015; Accepted November 20, 2015

ABSTRACT

The Reactome Knowledgebase (www.reactome.org) provides molecular details of signal transduction, transport, DNA replication, metabolism and other cellular processes as an ordered network of molecular transformations—an extended version of a classic metabolic map, in a single consistent data model. Reactome functions both as an archive of biological processes and as a tool for discovering unexpected functional relationships in data such as gene expression pattern surveys or somatic mutation catalogues from tumour cells. Over the last two years we re-developed major components of the Reactome web interface to improve usability, responsiveness and data visualization. A new pathway diagram viewer provides a faster, clearer interface and smooth zooming from the entire reaction network to the details of individual reactions. Tool performance for analysis of user datasets has been substantially improved, now generating detailed results for genome-wide expression datasets within seconds. The analysis module can now be accessed through a RESTful interface, facilitating its inclusion in third party applications. A new overview module allows the visualization of analysis results on a genome-wide Reactome pathway hierarchy using a single screen page. The search interface now provides auto-completion as well as a faceted search to narrow result lists efficiently.

INTRODUCTION

At the cellular level, life is a network of molecular reactions that include signal transduction, transport, DNA replication, protein synthesis and intermediary metabolism. In Reactome, these processes are systematically described in molecular detail to generate an ordered network of molecular transformations, resulting in an extended version of a classic metabolic map described by a single, consistent data model (1). The Reactome Knowledgebase thus systematically links human proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for discovering unexpected functional relationships in data such as gene expression pattern surveys or somatic mutation catalogues from tumour cells.

Since its inception 12 years ago, Reactome has grown to include (version 54—September 2015) entries for 8701 human genes (43% of the 20 296 predicted human protein-coding genes—http://Jul2015.archive.ensembl.org/Homo_sapiens/Info/Annotation), supporting the annotation of 18 658 specific forms of proteins distinguished by co- and post-translational modifications and subcellular localizations. These entities function together with 1540 small molecules as substrates, catalysts and regulators in 8770 reactions annotated on the basis of data from 20 708 literature references. These tallies include 1155 mutant variants and their post-translationally modified forms derived from 249 gene products, used to annotate 787 disease-specific reactions, tagged with 262 Disease Ontology terms (2). Recent additions include hedgehog signalling, host cell damage by

*To whom correspondence should be addressed. Tel: +1 212 263 5779; Fax: +1 212 263 8166; Email: deustp01@med.nyu.edu
Correspondence may also be addressed to Henning Hermjakob. Tel: +44 1223 494 671; Fax: +44 1223 494 468; Email: hhe@ebi.ac.uk
Correspondence may also be addressed to Lincoln Stein. Tel: +1 416 673 8514; Email: Lincoln.Stein@oicr.on.ca

© The Author(s) 2015. Published by Oxford University Press on behalf of Nucleic Acids Research.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

D482 *Nucleic Acids Research*, 2016, Vol. 44, Database issue

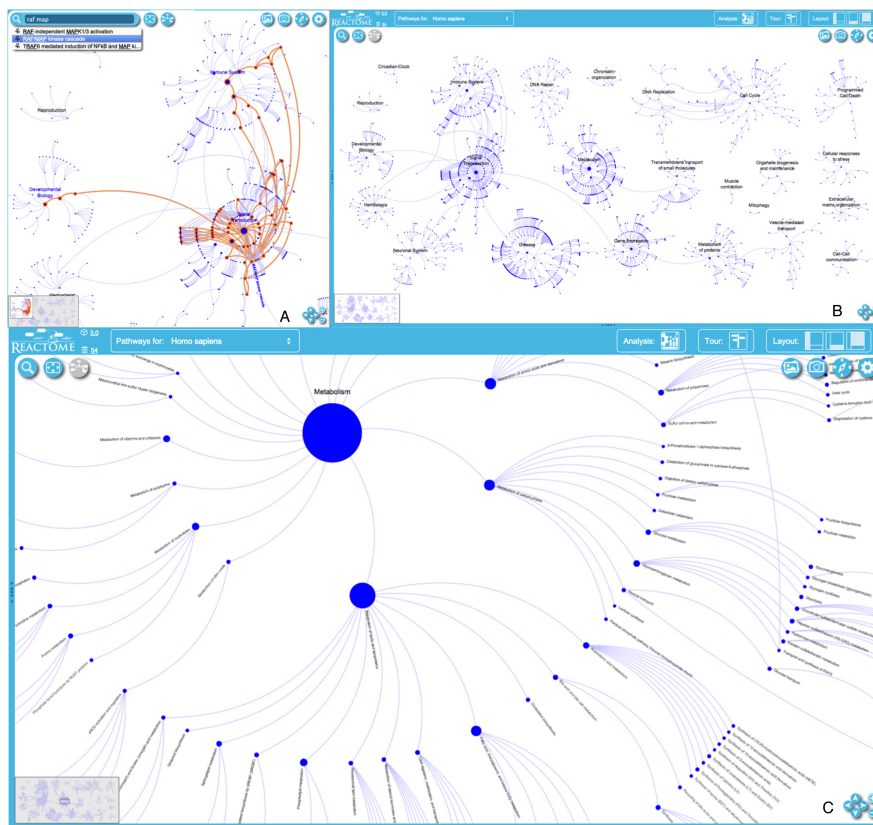


Figure 1. Pathway Overview. The entire pathways overview map (A). The RAF/MAP kinase cascade pathway is highlighted to show its involvement in multiple bursts (B). A zoomed-in view of the Metabolism burst showing individual subpathway groups (C).

bacterial toxins and extended annotations of DNA repair processes.

Here, we focus on three aspects of Reactome that have been extensively redesigned and improved since its last review in NAR (1): the web visualization and navigation browser, the toolkit for data analysis and the search utility.

PATHWAY OVERVIEW

Pathways in Reactome are organized hierarchically, grouping detailed pathways for translation, protein folding and post-translational modification into larger domains of biological function like protein metabolism. This hierarchical organization largely follows that of the Gene Ontology

(GO) biological process hierarchy (3,4). Reactome thus implements a pathway graph.

The pathway overview visualization provides an overview of all Reactome pathways, that highlights parent-child relationships and processes that are shared between pathways (Figure 1; <http://www.reactome.org/PathwayBrowser/>). In this view the 24 major Reactome pathway groups are each organized as a roughly circular 'burst'. The central node of each burst corresponds to the uppermost level of the Reactome event hierarchy (e.g. hemostasis, gene expression, signal transduction). Concentric rings of nodes around the central node represent successive more specific

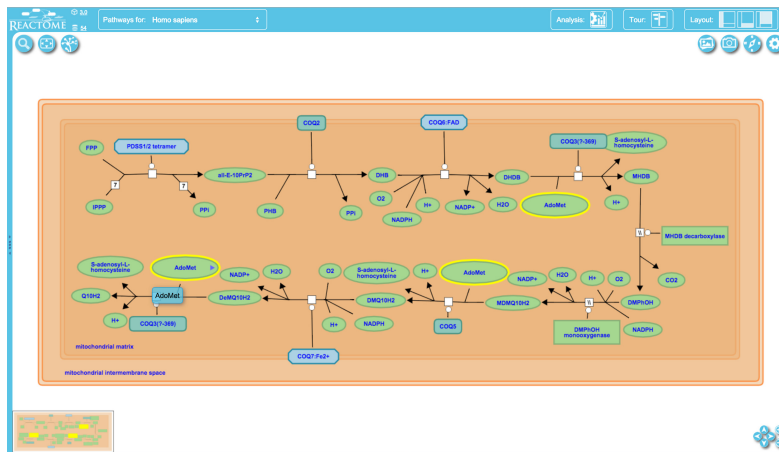


Figure 2. Diagram viewer. The central panel shows details of reactions and participating molecules in the nine-step process of ubiquinol (ubiquinol-10, Q10H2) biosynthesis. Buttons around the panel support functions including panning and zooming (lower right), changing the view (upper left) and downloading a snapshot of the pathway (upper right).

levels of the event hierarchy (e.g. signal transduction → signalling by FGFR → signalling by FGFR1). The arcs connecting nodes between successive rings within a burst represent parent-child (is-a) relationships in the event hierarchy. When a specific pathway like RAF/MAP kinase cascade is shared by more than one burst, arcs connect its nodes between bursts. A node's size is proportional to the number of physical entities (proteins, complexes, chemicals) it contains. Bursts are manually positioned to minimize crossing of arcs between bursts, and new bursts are manually added to the layout. With each new data release, a layout algorithm automatically adjusts the locations of existing nodes within the bursts to accommodate newly added nodes, maintaining spacing within rings and avoiding overlaps of nodes from neighbouring bursts, while minimizing displacement of the groups from their previous positions in the overview. Changes in the overall organization of the whole reaction network due to updates are thereby minimized, helping users identify and track areas of interest. This layout provides a legible, stable, informative overview and entry point to Reactome content even as the number of annotated proteins and processes in Reactome continues to increase.

DIAGRAM VIEWER

The new version of the diagram viewer reduces the loading time for diagrams and data, as well as the analysis results displayed on top of them. It provides visual feedback for common actions like hovering and focusing, has smoother transitions for zooming and selection and implements a mechanism to coordinate the amount of detail shown with

the zoom level—as the user zooms into specific parts of a diagram, more detailed information is progressively overlaid. A new search tool enables users to find items of interest within a diagram.

To support efficient navigation and searching within diagrams we have implemented a directed graph data structure which holds information such as the identities of the physical entities that make up complexes or sets and annotated preceding/following relationships between reactions in a pathway. This data structure is linked to the entities and events displayed in the diagram and takes advantage of graph traversing algorithms to support features such as rapid drilling down into complexes to reveal their components and navigation to all occurrences of an entity, both as an individual entity or as part of a larger composite entity, when present multiple times in a diagram (e.g. pyrophosphate (PPi) and H⁺ in Figure 2).

PATHWAY BROWSER

The pathway browser (<http://www.reactome.org/PathwayBrowser/>) (Figure 3) has been updated to reduce its loading time and provide a more attractive user interface. Buttons for widely used actions have been made more prominent, icons and colour schemes have been re-designed, and features including colour profiles can be customized by users. The pathway browser opens with the 'starburst' overview explained in the previous section. This overview is integrated with a diagram viewer that shows molecular details of pathways and individual reactions. When the pathway browser is loaded, the events hierarchy and the details panel appear on the left and bottom of the

D484 *Nucleic Acids Research*, 2016, Vol. 44, Database issue

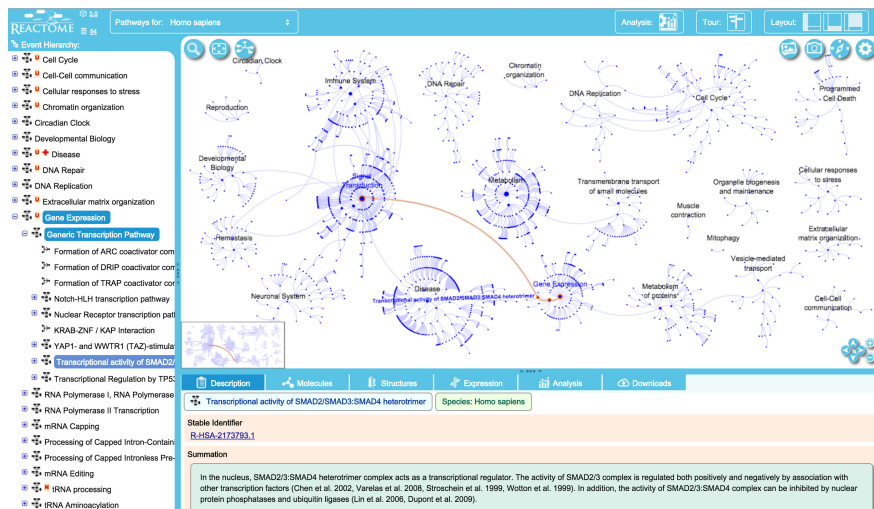


Figure 3. Pathway browser view centred on the ‘gene expression’ top-level pathway. Access to subpathways is provided via the hierarchical display of events on the left and by clicking on event nodes in the pathway display (viewport). Details for the selected event are shown in the panel under the pathway display. Buttons at the right of the top bar show the current version of our software (3.0) with access to our Github software repository, and the current version of our data (release 54). A button in the top bar provide access to the analysis tools (see below, Figure 4). Clicking on the layout buttons closes and re-opens the hierarchical display and details panels. The ‘tour’ button provides access to a brief video tour of the main features of the web site. Clicking on the gearwheel icon in the upper right corner of the pathway diagram provides access to a tool to customize diagram colouring and to an ‘About ...’ pop-up that briefly describes pathway diagram features and contains a link to the detailed users’ guide. (This guide is also accessible via the ‘documentation’ drop-down menu at the top of the home page).

viewport, respectively. The pathways overview widget is placed in the main viewport. Double clicking a pathway in the events hierarchy or its node in the main viewport will trigger a smooth, animated zoom in the main viewport to reveal the diagram for the pathway.

All display components are tightly connected, so that actions in one component will cause updates in others to consistently present information across the different display elements in accordance with the user’s selection. For example, choosing a reaction node or a physical entity glyph in the pathway diagram will trigger an update of the information displayed in the details panel under the pathway diagram and the events hierarchy panel on the left.

PATHWAY ANALYSIS

Reactome’s annotated data are a part of a list that shows what could happen if all annotated proteins and small molecules were present and active simultaneously in a cell. By overlaying an experimental dataset on these annotations, such as a list of genes activated in response to an experimental stimulus or expressed in transformed cells but not their normal counterparts, a user can search for patterns in the dataset such as modulation of specific pathways. By overlaying quantitative expression data or time series, a user can

visualize the extent of change in affected pathways and its progression.

Changing use patterns and growing data content are rapidly increasing performance demands for Reactome Pathway Analysis; high-throughput datasets often contain thousands or tens of thousands of identifiers. To address this challenge, we have re-implemented the analysis system, which now achieves interactive speed for genome-wide datasets, typically providing results for a dataset with 20 000 identifiers in less than 3 s. In addition to high execution speed, we now offer fine-grained results across all pathway levels in the Reactome events hierarchy. We provide a measure of target pathway coverage not only in terms of identified molecules, but also in terms of hit reactions per pathway.

The pathway analysis data submission interface is launched by selecting the analysis button located in the right top corner of the pathway browser. Once the user data is submitted by uploading or pasting a file into the allocated text area (Figure 4), the analysis is performed on the server side with the results shown in the pathway browser.

A new details panel displays results in tabular form. We have taken advantage of the new Reactome pathway overview visualization to show the analysis results as an overlay, allowing users to start with a high-level overview

D486 *Nucleic Acids Research*, 2016, Vol. 44, Database issue

The screenshot shows the Reactome website's search results page for the query 'raf map'. The page features a blue header with the Reactome logo and navigation links. A search bar at the top right contains the query 'raf map kina' and a search button. Below the search bar, a dropdown menu lists suggestions: 'raf map kinase', 'raf map kinase-3', 'raf map2k1 kinase', 'raf1 map kinase', 'raf map kinase-activated', 'raf map2k1 kinase-3', 'raf map2k7 kinase', 'raf1 map kinase-3', 'raf1 map2k1 kinase', and 'raf1-201 map kinase'. The main content area is titled 'Search results for raf map' and shows 'Showing 30 of 409' results. On the left, there are filter sections for 'Species' (with 'Homo sapiens' checked), 'Types' (with 'Reaction' checked), 'Compartments' (with 'cytosol' checked), and 'Reaction types' (with 'binds' checked). The search results are grouped into sections: 'Pathway' (5 results), 'Reaction' (5 results), and 'Other' (5 results). Each result is a link to a specific pathway or reaction, with the search terms 'raf' and 'map' highlighted in blue. For example, the first pathway result is 'RAF/MAP kinase cascade (Homo sapiens)' with a description: 'The RAS-RAF-MEK-ERK pathway regulates processes such as P... senescence'. The first reaction result is 'Raf activation (Homo sapiens)' with a description: 'Mammals have three RAF isoforms, A, B and C, that are activated downstream of RAS and stimulate the MAPK...'. The first other result is 'MAP kinase activation in TLR cascade (Homo sapiens)' with a description: 'The mitogen activated protein kinase (MAPK) cascade, one of the most ancient and evolutionarily conserved...'. The first reaction result is 'Raf activation (Homo sapiens)' with a description: 'Raf is a downstream effector of ras. Raf is activated upon phosphorylation at S338, oligomerization and membrane...'. The first other result is 'RAF phosphorylates MAP2K dimer (Homo sapiens)' with a description: 'Activated RAF phosphorylates the MEK kinases MAP2K1 and MAP2K2 on 2 serine residues in the MAP2K activation...'. The first reaction result is 'MAP2Ks and MAPKs bind to the activated RAF complex (Homo sapiens)'.

Figure 6. Redesigned search interface, showing term auto suggestion, grouping of results and highlighting of search terms in the results. The check boxes along the left side of the results page allow results to be further limited by species, data type, subcellular location and other parameters.

allows use of the Reactome server for batch dataset analysis. Over-representation and expression data analysis can be performed against the Reactome database (*/identifier* and */identifiers* methods) as well as species comparison (*/species* method). Once the data analysis or species comparison has been performed, a *token* is included in the client results allowing further service calls to refine the initial findings (*/token* and */download* methods).

FULL-TEXT SEARCH

The search tool has been redesigned to provide fast data access and incorporate additional data type attributes, yielding more accurate search results (Figure 6). The search core employs Solr, a high performance scalable full-text search engine specifically designed to search through large datasets. New features include filtering, results grouping, hit highlighting, spell checking and auto completion as the user types terms into the search text box.

CONCLUSIONS

The changes to the Reactome site and data analysis tools described here provide users with faster, easier access to Reactome

data increasing its utility both as an archive of known human biology and as a tool for generating and testing experimental hypotheses. The newly developed tools scale well to support the continued growth of Reactome content and its extension to new data types such as non-coding RNAs. These tools have been designed to support persistent growth in the number, size and complexity of user-supplied datasets for analysis.

ACKNOWLEDGEMENT

We are grateful to Ewan Birney for his advice and support, and to the many expert scientists who have collaborated with us as external authors and reviewers of Reactome content.

FUNDING

National Human Genome Research Institute at the National Institutes of Health [U41 HG003751; BD2K grant [U54 GM114833]; Ontario Research (GL2) Fund; European Bioinformatics Institute (EBI); Centre for Therapeutic Target Validation (CTTV). Funding for open access charge: National Institutes of Health [U41 HG003751].

Conflict of interest statement. None declared.

REFERENCES

1. Croft,D., Fabregat,A., Haw,R., Milacic,M., Weiser,J., Wu,G., Caudy,M., Garapati,P., Gillespie,M., Kamdar,M.R. *et al.* (2014) The Reactome Pathway Knowledgebase. *Nucleic Acids Res.*, **4**, D472–D477.
2. Kibbe,W.A., Arze,C., Felix,V., Mitraka,E., Bolton,E., Fu,G., Mungall,C.J., Binder,J.X., Malone,J., Vasant,D. *et al.* (2015) Disease Ontology 2015 update: an expanded and updated database of human diseases for linking biomedical knowledge through disease data. *Nucleic Acids Res.*, **43**, D1071–D1078.
3. Gene Ontology Consortium. (2000) Gene ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–29.
4. Gene Ontology Consortium. (2015) Gene Ontology Consortium: going forward. *Nucleic Acids Res.*, **43**, D1049–D1056.
5. Wijten,P., van Holten,T., Woo,L.L., Bleijerveld,O.B., Roest,M., Heck,A.J. and Scholten,A. (2013) High precision platelet releasate definition by quantitative reversed protein profiling—brief report. *Arterioscler. Thromb. Vasc. Biol.*, **33**, 1635–1638.

P.7. THE REACTOME PATHWAY KNOWLEDGEBASE (NAR 2013)

D472–D477 *Nucleic Acids Research*, 2014, Vol. 42, Database issue
doi:10.1093/nar/gkt1102

Published online 15 November 2013

The Reactome pathway knowledgebase

David Croft¹, Antonio Fabregat Mundo¹, Robin Haw², Marija Milacic², Joel Weiser², Guanming Wu², Michael Caudy², Phani Garapati¹, Marc Gillespie³, Maulik R. Kamdar², Bijay Jassal², Steven Jupe¹, Lisa Matthews⁴, Bruce May², Stanislav Palatnik², Karen Rothfels², Veronica Shamovsky⁴, Heeyeon Song², Mark Williams¹, Ewan Birney¹, Henning Hermjakob^{1,*}, Lincoln Stein^{2,5,6,*} and Peter D'Eustachio^{4,*}

¹European Bioinformatics Institute (EMBL-EBI), European Molecular Biology Laboratory, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK, ²Ontario Institute for Cancer Research, Toronto, ON M5G0A3, Canada, ³College of Pharmacy and Health Sciences, St. John's University, Queens, NY 11439, USA, ⁴NYU School of Medicine, New York, NY 10016, USA, ⁵Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11724, USA and ⁶Department of Molecular Genetics, University of Toronto, Toronto, ON M5S 1A1, Canada

Received October 1, 2013; Accepted October 19, 2013

ABSTRACT

Reactome (<http://www.reactome.org>) is a manually curated open-source open-data resource of human pathways and reactions. The current version 46 describes 7088 human proteins (34% of the predicted human proteome), participating in 6744 reactions based on data extracted from 15 107 research publications with PubMed links. The Reactome Web site and analysis tool set have been completely redesigned to increase speed, flexibility and user friendliness. The data model has been extended to support annotation of disease processes due to infectious agents and to mutation.

INTRODUCTION

At the cellular level, life is a network of molecular reactions that can be organized into higher order interconnected pathways. Molecules are synthesized, degraded, transported from one location to another and assembled into complexes and higher order structures with other molecules. Intensive studies of cellular signaling, motility, vesicular trafficking and other aspects of cell biology, coupled with the development of comprehensive catalogs of human genes and their protein products, have enabled the description of many cellular processes in the same molecular detail that has been a standard for metabolic processes for a generation. By annotating all of these processes in a single, consistent reaction-pathway format, the Reactome Knowledgebase systematically links human

proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for discovering unexpected functional relationships in data from gene expression pattern surveys or somatic mutation catalogues from tumor cells (e.g. 1–3).

Since its inception 10 years ago, Reactome has grown to include (version 46–September 2013) annotations for 7088 of the 20 774 protein-coding genes in the current Ensembl human genome assembly (34% coverage), 15 107 literature references and 1421 small molecules organized into 6744 reactions collected in 1481 pathways. Notable recent additions include extensive annotations of phospholipid and eicosanoid metabolism, protein glycosylation and SCF-KIT, IGF1R, NOTCH and HIPPO signaling, as well as annotations of regulatory processes mediated by non-coding RNAs.

Here, we will focus on three new features of Reactome: the development of a strategy to annotate the disease counterparts of normal human processes, the deployment of a redesigned Web site and the extension of tools for data analysis.

Disease curation using an enhanced data model

The Reactome data model (4) builds on earlier work by Kanehisa *et al.* (5) and Karp *et al.* (6) to classify and catalog physical entities (proteins and other macromolecules, small molecules, complexes of these entities and post-translationally modified forms of them), their subcellular locations and the transformations they can undergo (biochemical reaction, association to form a complex and translocation from one cellular compartment

*To whom correspondence should be addressed. Tel: +1 212 263 5779; Fax: +1 212 263 8166; Email: Peter.D'Eustachio@nyumc.org
Correspondence may also be addressed to Henning Hermjakob. Tel: +44 1223 494 671; Fax: +44 1223 494 468; Email: hhc@ebi.ac.uk
Correspondence may also be addressed to Lincoln Stein. Tel: +1 416 673 8514; Email: Lincoln.Stein@oicr.on.ca

The authors wish it to be known that, in their opinion, the first six authors should be regarded as Joint First Authors.

© The Author(s) 2013. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

to another). The central class of the Reactome data model is a *Reaction*, and subclasses of *Reaction* model these core biological events. Reactions are grouped into pathways, which in turn are assembled into a hierarchy of biological processes. Wherever appropriate, Reactome entities are linked to external reference databases such as UniProt (7), Ensembl (8), ChEBI (9) and Rhea (10). The full Reactome database schema is available at http://www.reactome.org/cgi-bin/classbrowser?DB=gk_current.

A straightforward definition of disease allows us to annotate a broad range of major disease processes at the molecular level. Diseases that can now be annotated in Reactome arise in one of three ways: a mutation, somatic or germ-line, leads to a non-functional gene product so processes that normally depend on that gene product do not take place; a mutation leads to a gene product with a novel function, enabling novel reactions whose products perturb normal human processes; or an infectious agent such as a virus introduces novel gene products whose novel reactions perturb normal human processes.

To identify disease-associated entities and events, a new 'disease' attribute is added, taking its value terms from a disease ontology [currently <http://disease-ontology.org/> (11)]. This attribute is multivalued, so an entity or event that has roles in multiple disease processes can be annotated to capture all of those roles.

With this addition, diseases due to infection can be annotated within our data structure. Our existing 'species' attribute allows pathogen-derived proteins, DNA and RNA to be distinguished from molecules encoded in the human genome. The 'species' attribute, also associated with complexes, reactions and pathways, can be multivalued, allowing complexes containing both host and viral proteins or reactions involving host and viral components to be properly identified. Finally, addition of Gene Ontology (GO) (12) host_cell terms to our cell compartment vocabulary allows us to localize pathogen-derived entities accurately, in compliance with GO annotation practice.

To denote molecular properties of a protein modified by a somatic or germline mutation in the gene that encodes it, as opposed to post translational modification, we created a new class. This class of genetic modifications has subclasses to accommodate substitution of a canonical residue by a different one, the insertion or deletion of multiple contiguous residues into a canonical sequence and the generation of a fusion protein containing fragments of two canonical ones (13). The underlying chemical similarity between a protein that differs from its canonical form due to a mutation and one that differs due to co- or post-translational modification allows us to maintain compliance with the PSI-MOD standard for annotation of protein modifications (14).

Reactions involving mutated proteins as catalysts, inputs, outputs and regulators are annotated exactly as wild-type reactions are. To link reactions involving a mutated protein to those involving its normal counterpart, an optional 'normal reaction' attribute is used. A reaction in which a receptor constitutively activated by a mutation transmits a signal is thereby paired with the wild-type reaction in which a normal receptor is activated by ligand binding. A reaction catalyzed by the normal form of an enzyme is paired with the different one catalyzed by its gain-of-function mutant counterpart or with a dead-end reaction (normal inputs, no outputs) associated with its loss-of-function mutant counterpart. A limitation of this strategy at present is that, as Reactome does not capture quantitative data such as reaction rates or binding affinities, quantitative effects of mutations are not readily annotated.

At the level of our event hierarchy, these normal-disease pairings support a disease event hierarchy that parallels our normal event hierarchy, a useful and generalizable organization. These pairings also support a visualization scheme that highlights the relationship between the normal and disease processes. As described previously (15), the physical entities and their interactions that

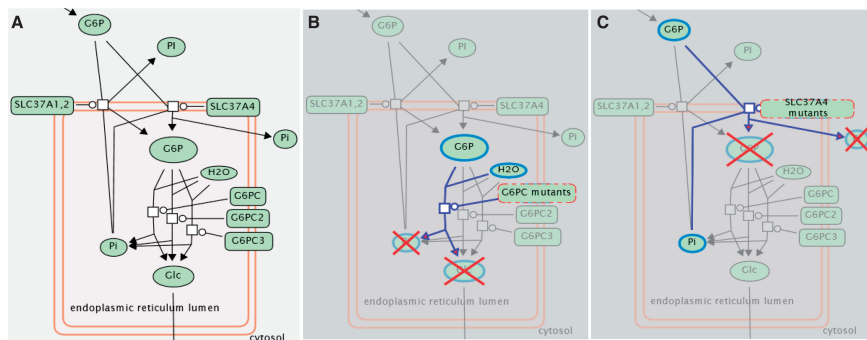


Figure 1. Example of disease curation and visualization in Reactome. The normal process of glucose export from the liver under fasting conditions (A) is disrupted by mutations that block glucose-6-phosphate hydrolysis within the endoplasmic reticulum (B) or the transport of glucose-6-phosphate and orthophosphate (Pi) between the endoplasmic reticulum and the cytosol (C).

D474 *Nucleic Acids Research*, 2014, Vol. 42, Database issue

comprise a pathway are laid out in a pathway diagram that follows the SBGN process description language (<http://www.sbgm.org/Documents/Specifications>) and that is displayed on our Web site. The software that generates these displays has been extended to create disease displays in which the variant forms of the events responsible for a disease process are superimposed and highlighted on the normal process diagram (Figure 1).

We have used these extensions of the Reactome data model and pathway visualization process to create a new disease pathway classification in our event hierarchy that incorporates existing material such as the HIV and influenza life cycles, amyloid formation and botulinus toxin neurotoxicity, together with new material that includes malignant transformation due to mutations in the EGFR and FGFR signaling pathways and mucopolysaccharidoses. Our current release includes annotations for 420 mutant forms of 68 proteins.

Updated Reactome Web site

Our home page has been redesigned completely to support intuitive access to our pathway browsing and data analysis tools. The new Web site retains a top menu bar to provide easy access to all of our tools and resources, accompanied by a central panel of links to our most widely used tools and a footer that displays all tools and resources. This organization is consistent with the general model for resources associated with EBI. News is now available as an interactive Twitter display, which also provides open real-time feedback, both by the Reactome group and our user community.

Dynamic pathway portal

We have improved the flexibility and performance of our pathway browser by creating a new pathway diagram visualization tool using the canvas element introduced in HTML 5. The canvas element is used to render whole pathway diagrams in a Google map-like way with XML-encoded pathway diagram data retrieved from the server using a RESTful API (Figure 2A). The new diagram visualization tool offers quicker performance and better data overlaying technologies (see later in text) and bypasses the slow step to generate static images during database release.

The event hierarchy panel to the left has been redesigned to provide interactivity and access to the entire listing of all the Reactome pathways. Icons now indicate whether a pathway is new (N) or updated (U), and identify ones that are parts of disease processes (+). Navigation controls in the upper left corner enable zooming and panning across the pathway panel. A diagram thumbnail in the lower left shows the part of the pathway currently displayed in the visualization panel. A widget icon in the upper right corner of the panel links to tools for searching within the displayed pathway, overlaying the pathway with functional interactors (see later in text) and for downloading the diagram as a snapshot or PNG file.

A new tabbed 'Details' panel, below the pathway diagram, provides additional graphical and textual information. The 'Overview' tab provides summary

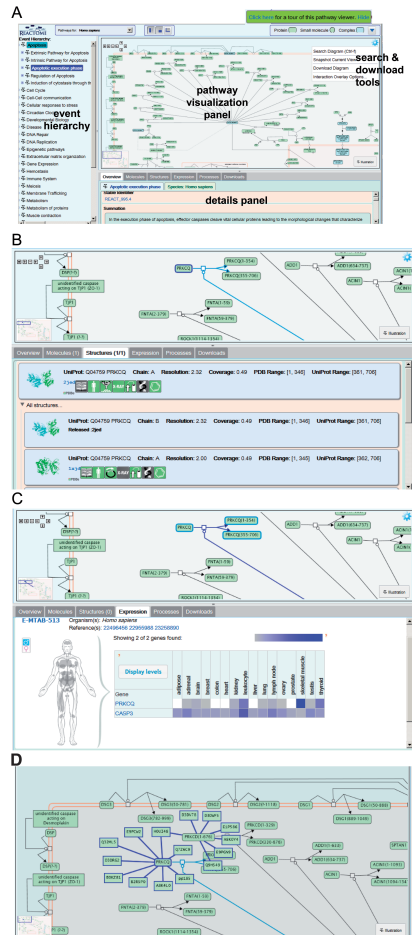


Figure 2. Visualization of the execution phase of apoptosis. (A) Choosing the entry for this event from the event hierarchy in the left panel of the web page causes the pathway to be displayed in a large panel to the right, laid out in SBGN process description format. Buttons at the top of the panel open a diagram key and give access to a brief tour of features of the web page. The panel at the bottom of the page contains text descriptions of various features of the pathway and participating molecules, linked to external databases. (B) The user has selected the reaction 'Caspase 3-mediated cleavage of PKC delta'; the 'Structures' detail tab displays PDB 3D structural data and citations for proteins in that reaction and the 'Expression' detail tab (C) displays condition-specific gene expression data from the Gene Expression Atlas. (D) The molecular interaction (MI) overlay displays proteins that interact with user-specified proteins PRKCO and PRKCD.

D476 *Nucleic Acids Research*, 2014, Vol. 42, Database issue

number of participating entities, relations between the pathways, average expression levels and other details.

The pathway names in the pathway-oriented view are clickable links, which take the user to the relevant pathway diagram. In it, the sub-pathway nodes, complexes and entities are colored according to expression level. Entities that lack gene expression information are colored white. Pathway and complex nodes are colored in vertical segments with each segment representing a protein in the pathway or a component of the complex. The segments are stacked in order from lowest to highest expression level. A new icon at the base of the diagram allows the user to single step through the individual experiments or time points.

Community relationships and data exchange

Reactome continues to collaborate with other data resources, such as GO, NCBI, EBI and WikiPathways (22). Reactome provides a series of link-outs to many online bioinformatics resources from its protein pages; we have added links to GeneCards annotations (23). Reactome is open-source and open-data, and we have continuously supported the major open-data standards in the domain, including BioPAX levels 2 and 3 (24), PSI MITAB (18), Protégé (<http://protege.stanford.edu>), SBML-ML (25) and SBGN export format. Reactome now provides an SBGN file format generated using libSBGN for individual pathways. The Reactome SBML export has been upgraded to Level 2, Version 4 and is enriched with a wide variety of additional annotations, including Systems Biology Ontology terms (26). Reactome also supports the Protein Ontology in developing an ontology for protein modifications and protein complexes (27). Our new RESTful API provides outside users with direct access to pathway data in Reactome.

The Reactome data model has been adopted by the Gramene group for manual annotation of plant pathways, especially metabolic processes specific to plants. The current release of Plant Reactome (<http://plants.reactome.org>) includes 131 rice pathways.

Since 2011, Reactome has participated in the Google Summer of Code program as part of the Genome Informatics group, helping to create the software components for the new pathway browser, RESTful API and the pathway overview. The Reactome data and source code continues to be publically accessible under the terms of a Creative Commons Attribution 3.0 Unported License.

ACKNOWLEDGEMENTS

The authors are grateful to the many expert scientists who have collaborated with us as external authors and reviewers of Reactome content. They thank three anonymous reviewers for their comments on this article.

FUNDING

National Human Genome Research Institute at the National Institutes of Health [U41 HG003751]; Ontario Research (GL2) fund; European Bioinformatics Institute;

European Commission (PSIMEx); Google Summer of Code Program (2011–2013). Funding for open access charge: National Institutes of Health [U41 HG003751].

Conflict of interest statement. None declared.

REFERENCES

- Biankin, A.V., Waddell, N., Kassahn, K.S., Gingras, M.C., Muthuswamy, L.B., Johns, A.L., Miller, D.K., Wilson, P.J., Patch, A.M., Wu, J. *et al.* (2012) Pancreatic cancer genomes reveal aberrations in axon guidance pathway genes. *Nature*, **491**, 399–405.
- Gieger, C., Radhakrishnan, A., Cvejic, A., Tang, W., Porcu, E., Pistis, G., Serbanovic-Canic, J., Elling, U., Goodall, A.H., Labrune, Y. *et al.* (2011) New gene functions in megakaryopoiesis and platelet formation. *Nature*, **480**, 201–208.
- Wu, G., Feng, X. and Stein, L. (2010) A human functional protein interaction network and its application to cancer data analysis. *Genome Biol.*, **11**, R53.
- Vastrik, I., D'Eustachio, P., Schmidt, E., Gopinath, G., Croft, D., de Bono, B., Gillespie, M., Jassal, B., Lewis, S., Matthews, L. *et al.* (2007) Reactome: a knowledge base of biologic pathways and processes. *Genome Biol.*, **8**, R39.
- Kanehisa, M., Goto, S., Sato, Y., Furumichi, M. and Tanabe, M. (2012) KEGG for integration and interpretation of large-scale molecular datasets. *Nucleic Acids Res.*, **40**, D109–D114.
- Karp, P.D., Krummenacker, M., Paley, S. and Wagg, J. (1999) Integrated pathway-genome databases and their role in drug discovery. *Trends Biotechnol.*, **17**, 275–281.
- UniProt Consortium. (2012) Reorganizing the protein space at the universal protein resource (UniProt). *Nucleic Acids Res.*, **40**, D71–D75.
- Flicek, P., Amode, M.R., Barrell, D., Beal, K., Brent, S., Carvalho-Silva, D., Clapham, P., Coates, G., Fairley, S., Fitzgerald, S. *et al.* (2012) Ensembl 2012. *Nucleic Acids Res.*, **40**, D84–D90.
- Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M. *et al.* (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Res.*, **41**, D456–D463.
- Alcántara, R., Axelsen, K.B., Morgat, A., Belda, E., Coudert, E., Bridge, A., Cao, H., de Matos, P., Ennis, M., Turner, S. *et al.* (2012) Rhea—a manually curated resource of biochemical reactions. *Nucleic Acids Res.*, **40**, D754–D760.
- Schriml, L.M., Arze, C., Nadenlla, S., Chang, Y.W., Mazaitis, M., Felix, V., Feng, G. and Kibbe, W.A. (2012) Disease ontology: a backbone for disease semantic integration. *Nucleic Acids Res.*, **40**, D940–D946.
- Gene Ontology Consortium. (2010) The gene ontology in 2010: extensions and refinements. *Nucleic Acids Res.*, **38**, D331–D335.
- Milacic, M., Haw, R., Rothfels, K., Wu, G., Croft, D., Hermjakob, H., D'Eustachio, P. and Stein, L. (2012) Annotating cancer variants and anti-cancer therapeutics in reactome. *Cancers*, **4**, 1180–1211.
- Montecchi-Palazzi, L., Beavis, R., Binz, P.A., Chalkley, R.J., Cottrell, J., Creasy, D., Shofstahl, J., Seymour, S.L. and Garavelli, J.S. (2008) The PSI-MOD community standard for representation of protein modification data. *Nat. Biotechnol.*, **26**, 864–866.
- Croft, D., O'Kelly, G., Wu, G., Haw, R., Gillespie, M., Matthews, L., Caudy, M., Garapati, P., Gopinath, G., Jassal, B. *et al.* (2011) Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Res.*, **39**, D691–D697.
- Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. and Bourne, P.E. (2000) The protein data bank. *Nucleic Acids Res.*, **28**, 235–242.
- Kapusheky, M., Adamusiak, T., Burdett, T., Culhane, A., Farne, A., Filippov, A., Holloway, E., Klebanov, A., Kryvych, N., Kurbatova, N. *et al.* (2012) Gene expression atlas update - a value-added database of microarray and sequencing-based

- functional genomics experiments. *Nucleic Acids Res.* **40**, D1077–D1081.
18. Aranda, B., Blankenburg, H., Kerrien, S., Brinkman, F.S., Ceol, A., Chautard, E., Dana, J.M., De Las Rivas, J., Dumousseau, M., Galeota, E. *et al.* (2011) PSICQUIC and PSISCORE: accessing and scoring molecular interactions. *Nat. Methods*, **8**, 528–529.
19. Liu, T., Lin, Y., Wen, X., Jorissen, R.N. and Gilson, M.K. (2007) BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucleic Acids Res.* **35**, D198–D201.
20. Knox, C., Law, V., Jewison, T., Liu, P., Ly, S., Frolkis, A., Pon, A., Banco, K., Mak, C., Neveu, V. *et al.* (2011) DrugBank 3.0: a comprehensive resource for 'omics' research on drugs. *Nucleic Acids Res.* **39**, D1035–D1041.
21. Warde-Farley, D., Donaldson, S.L., Comes, O., Zuberi, K., Badrawi, R., Chao, P., Franz, M., Grouios, C., Kazi, F., Lopes, C.T. *et al.* (2010) The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function. *Nucleic Acids Res.* **38**, W214–W220.
22. Kelder, T., van Iersel, M.P., Hanspers, K., Kutmon, M., Konkin, B.R., Evelo, C.T. and Pico, A.R. (2012) WikiPathways: building research communities on biological pathways. *Nucleic Acids Res.* **40**, D1301–D1307.
23. Safran, M., Dalah, I., Alexander, J., Rosen, N., Iny Stein, T., Shmoish, M., Nativ, N., Bahir, I., Doniger, T., Krug, H. *et al.* (2010) GeneCards Version 3: the human gene integrator. *Database*, **2010**, baq020.
24. Demir, E., Cary, M.P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J. *et al.* (2010) The BioPAX community standard for pathway data sharing. *Nat. Biotechnol.* **28**, 935–942.
25. Bornstein, B.J., Keating, S.M., Jouraku, A. and Hucka, M. (2008) LibSBML: an API Library for SBML. *Bioinformatics*, **24**, 880–881.
26. Courtot, M., Juty, N., Knüpf, C., Waltemath, D., Zhukova, A., Dräger, A., Dumontier, M., Finney, A., Golebiewski, M., Hastings, J. *et al.* (2011) Model storage, exchange and integration. *Mol. Syst. Biol.* **7**, 543.
27. Natale, D.A., Arighi, C.N., Barker, W.C., Blake, J.A., Bult, C.J., Caudy, M., Drabkin, H.J., D'Eustachio, P., Evsikov, A.V., Huang, H. *et al.* (2011) The protein ontology: a structured representation of protein forms and complexes. *Nucleic Acids Res.* **39**, D539–D545.

O. PUBLICATIONS INDIRECTLY RELATED TO THE PRESENTED WORK

O.1. GRAMENE 2018: UNIFYING COMPARATIVE GENOMICS AND PATHWAY RESOURCES FOR PLANT RESEARCH

Nucleic Acids Research, gkx1111, <https://doi.org/10.1093/nar/gkx1111> (20 November 2017)

Abstract

Gramene (<http://www.gramene.org>) is a knowledgebase for comparative functional analysis in major crops and model plant species. The current release, #54, includes over 1.7 million genes from 44 reference genomes, most of which were organized into 62,367 gene families through orthologous and paralogous gene classification, whole-genome alignments, and synteny. Additional gene annotations include ontology-based protein structure and function; genetic, epigenetic, and phenotypic diversity; and pathway associations. Gramene's Plant Reactome provides a knowledgebase of cellular-level plant pathway networks. Specifically, it uses curated rice reference pathways to derive pathway projections for an additional 66 species based on gene orthology, and facilitates display of gene expression, gene-gene interactions, and user-defined omics data in the context of these pathways. As a community portal, Gramene integrates best-of-class software and infrastructure components including the Ensembl genome browser, Reactome Pathway Browser, and Expression Atlas widgets, and undergoes periodic data and software upgrades. Via powerful, intuitive search interfaces, users can easily query across various portals and interactively analyze search results by clicking on diverse features such as genomic context, highly augmented gene trees, gene expression anatomograms, associated pathways, and external informatics resources. All data in Gramene are accessible through both visual and programmatic interfaces.

O.2. OPEN TARGETS: A PLATFORM FOR THERAPEUTIC TARGET IDENTIFICATION AND VALIDATION

Nucleic Acids Research, Volume 45, Issue D1, 4 January 2017, Pages D985–D994, <https://doi.org/10.1093/nar/gkw1055> (08 December 2016)

Abstract

We have designed and developed a data integration and visualization platform that provides evidence about the association of known and potential drug targets with diseases. The platform is designed to support identification and prioritization of biological targets for follow-up. Each drug target is linked to a disease using integrated genome-wide data from a broad range of data sources. The platform provides either a target-centric workflow to identify diseases that may be associated with a specific target, or a disease-centric workflow to identify targets that may be associated with a specific disease. Users can easily transition between these target- and disease-centric workflows. The Open Targets Validation Platform is accessible at <https://www.targetvalidation.org>.

O.3. PLANT REACTOME: A RESOURCE FOR PLANT PATHWAYS AND COMPARATIVE ANALYSIS

Nucleic Acids Research, Volume 45, Issue D1, 4 January 2017, Pages D1029–D1039, <https://doi.org/10.1093/nar/gkw932> (30 October 2016)

Abstract

Plant Reactome (<http://plantreactome.gramene.org/>) is a free, open-source, curated plant pathway database portal, provided as part of the Gramene project. The database provides intuitive bioinformatics tools for the visualization, analysis and interpretation of pathway knowledge to support genome annotation, genome analysis, modeling, systems biology, basic research and education. Plant Reactome employs the structural framework of a plant cell to show metabolic, transport, genetic, developmental and signaling pathways. We manually curate molecular details of pathways in these domains for reference species *Oryza sativa* (rice) supported by published literature and annotation of well-characterized genes. Two hundred twenty-two rice pathways, 1025 reactions associated with 1173 proteins, 907 small molecules and 256 literature references have been curated to date. These reference annotations were used to project pathways for 62 model, crop and evolutionarily significant plant species based on gene homology. Database users can search and browse various components of the database, visualize curated baseline expression of pathway-associated genes provided by the Expression Atlas and upload and analyze their Omics datasets. The database also offers data access via Application Programming Interfaces (APIs) and in various standardized pathway formats, such as SBML and BioPAX.

O.4. GRAMENE 2016: COMPARATIVE PLANT GENOMICS AND PATHWAY RESOURCES

Nucleic Acids Research, Volume 44, Issue D1, 4 January 2016, Pages D1133–D1140, <https://doi.org/10.1093/nar/gkv1179> (08 November 2015)

Abstract

Gramene (<http://www.gramene.org>) is an online resource for comparative functional genomics in crops and model plant species. Its two main frameworks are genomes (collaboration with Ensembl Plants) and pathways (The Plant Reactome and archival BioCyc databases). Since our last NAR update, the database website adopted a new Drupal management platform. The genomes section features 39 fully assembled reference genomes that are integrated using ontology-based annotation and comparative analyses, and accessed through both visual and programmatic interfaces. Additional community data, such as genetic variation, expression and methylation, are also mapped for a subset of genomes. The Plant Reactome pathway portal (<http://plantreactome.gramene.org>) provides a reference resource for analyzing plant metabolic and regulatory pathways. In addition to ~200 curated rice reference pathways, the portal hosts gene homology-based pathway projections for 33 plant species. Both the genome and Pathway Browsers interface with the EMBL-EBI's Expression Atlas to enable the projection of baseline and differential expression data from curated expression studies in plants. Gramene's archive website (<http://archive.gramene.org>) continues to provide previously reported resources on comparative maps, markers and QTL. To further aid our users, we have also introduced a live monthly educational webinar series and a Gramene YouTube channel carrying video tutorials.

The aim of this research was to optimise the performance of the storage, retrieval, analysis and interactive visualisation of biomolecular pathways data. This was achieved by the adoption of new technologies and a variety of highly optimised data structures, algorithms and strategies across the different layers of the software.

The first challenge to overcome was the creation of a long-lasting, large-scale web application to enable pathways navigation; the Pathway Browser. This tool had to aggregate different modules to allow users to browse pathway content and use their own data to perform pathway analysis.

Another challenge was the development of a high-performance pathway analysis tool to enable the analysis of genome-wide datasets within seconds. Once developed, it was also integrated into the Pathway Browser allowing interactive exploration and analysis of high throughput data.

The Pathways Overview layout and widget were created to enable the representation of the complex parent-child relationships present in the pathways hierarchical organisation. This module provides a means to overlay analysis results in such a way that the user can easily distinguish the most significant areas of biology represented in their data. Although an existing force-directed layout algorithm was initially utilised for the graphical representation, it did not achieve the expected results and a custom radial layout algorithm was developed instead.

A new version of the pathway Diagram Viewer was engineered to achieve loading and rendering of 97% of the target diagrams in less than 1 second. Combining the multi-layer HTML5 Canvas strategy with a space partitioning data structure minimised CPU workload, enabling the introduction of new features that further enhance user experience.

On the server side, the work focused on the adoption of a graph database (Neo4j) and the creation of the new Content Service (REST API) that provides access to these data. The Neo4j graph database and its query language, Cypher, enabled efficient access to the complex pathway data model, facilitating easy traversal and knowledge discovery. The adoption of this technology greatly improved query efficiency, reducing the average query time by 93%.